Universiteit Leiden

Master Computer Science

Thesis: Benchmarking Reinforcement Learning Algorithms for Demand Response in Urban Energy Systems

Name:	Lennard Schaap
Student ID:	1914839
Date:	July 8, 2024
Specialisation:	Artificial Intelligence
Supervisors:	Thomas Moerland & Koen Ponse
Second Reader:	Aske Plaat

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Abstract

Increased energy usage in urban areas, combined with high peaks of electricity demand due to the integration of more variable energy sources like photovoltaic arrays, poses significant challenges to the modern power grid. The increased availability of distributed energy resources in residential buildings can be leveraged by smart controllers to facilitate demand response and flatten the energy demand curve. Reinforcement Learning (RL) emerges as a promising solution for continuous control problems in this context. In contrast to traditional rule-based controllers, RL approaches show superior performance in addressing demand response challenges. This research provides a detailed benchmarking study of state-of-the-art RL algorithms in environments of increasing complexity, characterised by a growing number of buildings in a district. Our findings indicate that, in contrast to earlier research, the DDPG algorithm outperforms other benchmarked algorithms across all district sizes. Additionally, this study reveals that a single-agent approach outperforms a multi-agent approach in smaller districts and performs comparably in larger districts. Although performance curves during training show better results for independent agents, their test-time performance is similar to the single-agent approach, suggesting that independent agents may overgeneralise on the training data. Overall, this research highlights the significant potential of RL in addressing the complexities of demand response in urban energy systems.

Acknowledgements

Thanks to my supervisors Koen Ponse and Thomas Moerland for guiding me through this project, supporting me, and providing valuable feedback. Thanks to Kingsley Nweye and Zoltan Nagy for providing me with the dataset used in this study, the source code to replicate their experiments and a preprint of the associated paper, while they were still not publicly available. Additionally, thanks to ALICE for providing a significant portion of the computation for the experiments conducted in this thesis. Lastly, I want to express my gratitude to LIACS for providing an excellent academic environment and support throughout my studies.

Contents

1 Introduction

The increase of electricity consumption and high peaks of electricity usage in urban areas, coupled with rising electricity prices, underscore the need for smart energy control systems. The modern power grid faces significant challenges due to the rising demand for electricity and the integration of more variable energy sources (Sinsel et al., 2020) (Rangu et al., 2020). One promising approach to address these challenges is the deployment of systems that leverage distributed energy resources such as batteries, thermal storage, photovoltaic panels, electric vehicles, heat pumps, and water heaters (Akorede et al., 2010) to optimise energy usage. By integrating these technologies, we can develop grid-interactive efficient buildings that dynamically respond to grid conditions. This integration supports the creation of systems that can adjust their energy consumption in response to grid signals, thereby enhancing overall grid stability.

To illustrate the issue of the high peaks mentioned, consider the adoption of renewable energy sources, particularly solar panels, which introduces significant fluctuations into urban power grid energy demands. During sunny periods, communities with widespread solar panel use experience a steep decline in the demand for fossil-fuelled power. However, as the sun sets and solar generation decreases, there is a rapid surge in the demand for fossil-fuelled electricity to fill the gap. A similar pattern is observed with the rise of electric vehicles, which cause a major energy demand spike during charging periods, particularly in the evening.

This abrupt shift in energy demand, known as the "duck curve", poses a serious threat to power grid stability (Neukomm et al., 2019). Periods of high electricity demand in cities and districts drive up prices and overall power distribution costs, and can even result in blackouts. This phenomenon highlights the critical need for effective demand response strategies and energy storage solutions to maintain grid resiliency and balance renewable energy supply with fluctuating demand. These strategies aim to flatten and reduce the overall electrical demand curve.

Demand Response (DR) involves adjusting electricity usage in response to grid signals or market conditions, crucial for balancing supply and demand as renewable energy sources like solar and wind expand. One of the key challenges is aligning electricity demand with fluctuating renewable energy generation while maximising renewable energy usage (Gils, 2014). Energy storage systems (ESS) such as batteries and smart HVAC systems help mitigate generation-demand discrepancies (Tan et al., 2013). Efficient scheduling, leveraging ESS and HVAC as storage solutions, aids in flattening the energy demand curve by avoiding peak-hour power draws (Vazquez-Canteli et al., 2012). DR strategies include shifting non-essential electricity use to off-peak times, reducing consumption during high-demand periods, and temporarily shifting loads to alternative or stored energy sources. These measures not only enhance grid stability by mitigating the duck curve effect but also yield cost savings and environmental benefits through reduced reliance on fossil fuels during peak demand, promoting a more sustainable urban energy infrastructure.

Traditional building control methods used to address problems such as DR include heuristic strategies developed by domain experts (Nagy et al., 2023) such as Rule-Based Control (RBC). However, these control schemes may not be optimal as they are static and do not adapt to specific building conditions. Adaptive control strategies, such as model predictive control (MPC), have also been introduced to overcome these limitations. MPC uses a model to forecast the impact of future inputs on building dynamics using a predefined accurate physical model of the system. The need for such models in MPC makes the DR response problem addressed in this work particularly challenging, as every commercial building is unique and necessitates expert knowledge to develop precise building models (Nagy et al., 2023). Moreover, employing MPC can be computationally expensive due to the need to solve optimisation problems online (Cao et al., 2020).

Reinforcement Learning (RL) emerges as a promising approach for addressing the complexities of DR due to its ability to learn optimal control strategies in dynamic and uncertain environments (François-Lavet et al., 2016). Unlike traditional rule-based methods, RL algorithms can adapt and improve their decision-making processes over time through interaction with the environment. This adaptability is crucial in DR contexts where grid conditions, renewable energy generation, and consumer demand patterns can vary unpredictably. In this unpredictable setting, RL agents learn from data generated by the environment and can recognise patterns that might not be apparent to human experts. In recent years, RL has

achieved significant milestones in solving complex decision-making problems, including surpassing human performance in games like Atari (Mnih et al., 2013), Go (Silver et al., 2016) (Silver et al., 2017), and StarCraft II (Vinyals et al., 2019), as well as mastering control tasks in robotics (Kalashnikov et al., 2018) (Haarnoja et al., 2024) and other continuous control tasks such as controlling cooling systems (Luo et al., 2022). In contrast to MPC, model-free RL methods learn by interacting with the environment without the need of a model of the building dynamics. Another benefit of RL is that optimisation happens offline, which eliminates the need of computationally expensive online optimisation used by methods such as MPC.

CityLearn (Vazquez-Canteli et al., 2012) enables simulation of districts with diverse loads, devices, and energy storage, offering a standardised method for modelling urban energy systems. This project uses CityLearn to benchmark RL algorithms for energy coordination and demand response. CityLearn serves as a robust framework for comparing algorithms to reshape demand curves and optimise DR strategies, using pre-simulated models for residential and commercial buildings. CityLearn is designed as a Gymnasium environment (Towers et al., 2022), making it possible to implement RL algorithms to improve DR.

CityLearn is widely used for benchmarking RL algorithms, featuring an annual challenge where participants compete to create the best algorithm for specified problems and datasets. These datasets are also referenced in various other research papers (Dhamankar et al., 2020) (Pinto et al., 2022). However, current research tends to focus on individual datasets, specialised methods and different optimisation goals, often neglecting direct comparisons of different algorithms on the same datasets.

The CityLearn environment enables two approaches to address the DR problem: utilising a central agent that controls all building energy systems or employing separate agents that control each building in the district separately. While some literature compares single-agent methods with centralised agent methods, these studies typically do not benchmark different algorithms against each other. Similarly, existing research on the comparison of central agent methods with independent multi-agent methods often examines only a single algorithm per approach.

Evidently, there is a notable gap in the literature concerning the benchmarking of RL methods across DR scenarios of varying complexities. This work aims to fill this gap by conducting a comprehensive benchmarking study of both single-agent and multi-agent RL approaches on DR problems of varying complexity. By gradually increasing the size of the district, we compare the performance of these RL approaches against a classic RBC baseline.

The main goal of this research is to address two primary research questions:

1. *How do different Reinforcement Learning algorithms compare in solving Demand Response problems of varying complexity in the CityLearn environment?*
2. *How do single-agent and multi-agent approaches compare in this context?*

In contrast to findings in existing literature that indicate that independent agents approaches work better than central agent approaches, this research indicates that when increasing the network size of the RL agents to accommodate the increasing complexity of the problem, central agent methods outperform independent multi-agent methods on smaller district sizes. Although the performance of the independent agents on the training data indicates significantly better results, their evaluation on the test data shows they perform comparably with central agent approaches across all evaluated KPIs for larger district sizes. Furthermore, our findings suggest that in both single-agent and multi-agent approaches, the DDPG algorithm consistently outperforms all benchmarked algorithms across all environment settings. The source code and trained models from this project are available at GitHub [1].

In Section 2, we will discuss related research that explores different RL approaches within grid-interactive buildings, focusing on optimisation objectives like DR. This includes comparisons between single-agent and multi-agent strategies. In Section 3, we will discuss the foundational concepts of RL, essential for understanding the methodologies used in this work. We will also frame the DR problem within the RL

[1] https://github.com/LennardSchaap/citylearn_benchmark

framework using CityLearn as the environment. In Section 4, we introduce the CityLearn environment, the CityLearn challenges and detail the specific dataset utilised in this study. This dataset enables comparisons of RL methods across various problem complexities, in contrast to the datasets introduced in the CityLearn challenges. In Section 6, we introduce the RL algorithms that will be evaluated both in a central agent setting and in an independent multi-agent setting. In Section 7, we present the main findings of this study, highlighting the comparative performance of the benchmarked RL algorithms in central agent and multi-agent settings, on different problem scales. In Section 9 we discuss the implications of our findings and outline future research directions that can be pursued in this field.

2 Related Work

In this section, we review existing research on RL control in urban energy systems, focusing on the challenges and proposed solutions. We highlight key studies that have used the CityLearn environment and similar frameworks to develop different approaches for solving them, including RL. Additionally, this overview examines different works comparing multi-agent with single-agent approaches. Finally, we discuss how our study addresses an unresolved problem by benchmarking different RL algorithms in increasingly complex environments.

2.1 Defining RL Control Challenges in Urban Energy Systems

Nweye et al. present a collection of nine challenges designed for RL control in grid-interactive buildings, establishing a standardised environment for evaluating RL approaches (Nweye et al., 2022a). They implement one of these challenges in CityLearn, using the dataset from the CityLearn Challenge 2021 (Nagy et al., 2021). They compare a Soft Actor-Critic algorithm with MARLISA (Vazquez-Canteli et al., 2020), a multi-agent RL algorithm that was developed in parallel with CityLearn, using different offline training periods based on a rule-based controller (RBC). They discover that extended offline training using the fixed logs from an optimised RBC results in improved long-term performance, albeit with a slower convergence.

2.2 Proposed Solutions

Chen et al. utilise the CityLearn environment with data from the CityLearn Challenge 2021, comprising simulation data from 9 buildings across 4 seasons over a year (Nagy et al., 2021) (Chen et al., 2020). Their approach uses sequence-to-sequence models for energy demand and solar generation forecasting, integrating weather and building information. Gated Recurrent Units are used for solar generation prediction. A central load aggregator, employing Natural Evolutionary Strategies, determines target loads for load-shifting via HVAC systems. Their findings suggest that their approach scores 16.8% better than the RBC baseline in terms of the average net electricity consumption, load factor, ramping, average daily peak demand, and annual peak demand KPIs.

Yao et al. utilise a micro-grid environment similar to CityLearn called Power Grid World (Biagioni et al., 2022) to connect multiple micro-grids with HVAC, PV, and ESS systems. They utilise the model-based MuZero algorithm framework for optimisation. The reward structure of the scheduling model includes components like discomfort reward, energy consumption reward, HVAC system reward, and system reward. They demonstrate the applicability of the MuZero algorithm to microgrid scheduling problems, but do not provide comparisons with other methods in their results (Schrittwieser et al., 2020).

Nweye et al. utilise the CityLearn 2022 dataset along with additional weather and carbon intensity data. They employ an independent SAC agent named MERLIN, which initially uses the RBC policy for action selection during exploration. The reward function aims to minimise both cost and carbon emissions, encouraging net-zero energy consumption and penalising net export when the battery is not fully charged. Their agent outperforms the baseline RBC by 15% in their experiments (Nweye et al., 2023c).

Kathirgamanathan et al. introduced a centralised SAC agent for the CityLearn environment using data from the CityLearn Challenge 2021, achieving an average improvement of 3.3% over the RBC baseline

(Kathirgamanathan et al., 2020). In their discussion, they highlight that the performance of the centralised SAC agent has not been evaluated across problems of varying scales, suggesting this as a potential area for future research.

2.3 Multi-Agent vs. Single-Agent Comparisons

Pinto et al. used a CityLearn environment with four buildings (a restaurant and three multi-family homes) connected to the grid to compare multi-agent SAC versus single-agent SAC (Pinto et al., 2022). They evaluated economic (cost), grid-interaction (peak demand, peak-to-average ratio, daily peak), and flexibility (self-sufficiency) KPIs. Both architectures reduced costs, peak demand, and improved self-sufficiency compared to the RBC baseline, with notable reductions in daily peak and PAR across various climates. The multi-agent approach slightly outperformed the single-agent approach. They suggest future work to compare these approaches on a larger number of buildings to better assess scalability and performance.

Almilaify et al. address the problem mentioned by Kathirgamanathan et al. by examining the scalability of an independent multi-agent SAC algorithm against a single-agent SAC algorithm by comparing them on different neighbourhood sizes, scaling from 2 to 64 (Almilaify et al., 2023). They found that the independent SAC agent performs better on every task while the central agent scored comparably with the RBC agent. While this study compares the SAC algorithm on different problem scales and hypothesises that the central agent approach does not scale well, it does not use this scalable problem setting as a benchmark for comparing different RL control algorithms.

2.4 Contributions

In contrast to earlier works, this research demonstrates that central agent methods outperform independent agent methods in smaller districts when the neural network size is increased. In larger problem sizes, the performance of both approaches is similar. This study finds that using the same network sizes for both central and independent agents introduces a challenge due to the central agents' observation space scaling with problem complexity. To mitigate this issue, we size the central agents' network to match the combined sizes of the independent agents' networks for fair comparison of the two approaches. Additionally, this study reveals that while the literature often favours SAC agents for CityLearn problems, our findings demonstrate that DDPG agents outperform SAC agents across all problem settings.

3 Preliminaries

In this section, we introduce the formal concepts of Reinforcement Learning (RL) that are essential for understanding the methods used in this work, which are detailed in Section 6. We frame the energy demand response problem as an RL challenge, utilising CityLearn as the environment in which our RL agent operates.

3.1 Reinforcement Learning

Reinforcement learning is a paradigm of machine learning that differs from fields such as supervised learning or unsupervised learning in that it does not require a labelled dataset. Instead, RL assumes an agent in an environment that interacts with the environment by taking actions $a \in \mathcal{A}$. The agent observes the outcome of its actions, transitioning to new states $s \in \mathcal{S}$ and taking subsequent actions, thus accumulating data. The initial state of the environment is denoted s_0, and the initial action taken by the agent is denoted a_0. In addition to transitioning to a new state, the agent receives a (positive or negative) 'reward', r, which indicates the desirability of the current state. The goal of the agent in the RL setting is to select actions that maximise the cumulative reward, known as the return G. Figure 1 illustrates the agent-environment loop in RL.

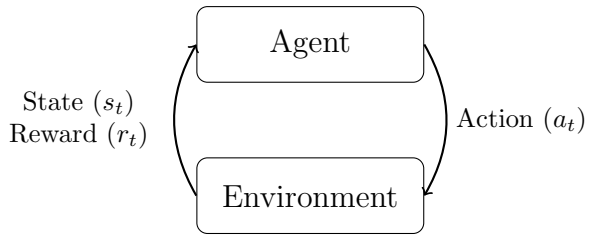


Figure 1: The agent-environment interaction loop in Reinforcement Learning. During training, the agent explores the environment by selecting actions, transitioning to new states that yield specific rewards. Post-training, the agent follows its learned policy to select actions.

3.1.1 Markov Decision Process

We can formalise the CityLearn environment as a Markov Decision Process (MDP), on which we apply RL for solving it. An MDP is a 6-tuple, $\langle \mathcal{S}, \mathcal{A}, T, R, \gamma, H \rangle$, where $\mathcal{S}$ is the set of all possible environment states, $\mathcal{A}$ is the set of possible actions the agent can take, $T : \mathcal{S} \times \mathcal{A} \to \mathcal{P}(\mathcal{S})$ is the transition function, where $P(s'|s, a) \in [0, 1]$ describes the probability of transitioning into state s' after taking action a in state s, $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the reward function, with $r_t = R(s_t, a_t, s_{t+1})$, $\gamma \in [0, 1]$ is the discount factor, which represents the difference in importance between future rewards and present rewards. The horizon H is the time span over which the agent interacts with the environment.

3.1.2 States and Actions

A state is the complete description of the environment. The observation (o) that the agent perceives is a subset of this state. When the agent can observe the entire state, the environment is fully observable. When the agent can only see a subset of the complete state, the environment is partially observable.

The environment can be either stochastic or deterministic. For example, in chess, each action (making a move) results in a deterministic new board state. In backgammon, the state depends on the roll of the dice, making the environment stochastic. The probability of ending up in a specific state in an environment is denoted as $P(s'|s, a)$ where s is the current state, s' is the next state and a is the action taken. In a deterministic environment, each action taken from a given state s always leads to the same next state s', resulting in a transition probability $P(s'|s, a) = 1$.

A series of actions taken by the agent in the environment up until a certain horizon H is called a trajectory. The trajectory can be represented as $\tau = (s_0, a_0, s_1, a_1, \ldots, s_{H-1}, a_{H-1}, s_H)$, where s_i is the state at time step i and a_i is the action taken at time step i. The probability of a specific trajectory is given by:

$$\Pr(\tau) = \Pr(s_0) \prod_{i=0}^{H-1} P(s_{i+1}|s_i, a_i) \Pr(a_i|s_i) \tag{1}$$

where $\Pr(s_0)$ is the probability of the initial state, $P(s_{i+1}|s_i, a_i)$ is the probability of transitioning to state s_{i+1} given the transition function T, state s_i and action a_i. $\Pr(a_i|s_i)$ is the probability of taking action a_i given state s_i.

3.1.3 Policies

The action the agent takes is determined by its policy, which can be stochastic or deterministic. A deterministic policy is often denoted as μ, where $a_t = \mu(s_t)$ and a stochastic policy is denoted by π, where $a_t \sim \pi(s_t)$. A stochastic policy is a function that maps states to probability distributions over actions, from which actions can then be sampled. The action space is defined as $\mathcal{A}$, which represents all possible actions that the agent can take in the environment. It encompasses both discrete and continuous action sets, depending on the nature of the task. In discrete action spaces, $\mathcal{A}$ is typically represented as a finite set of actions. For continuous action spaces, $\mathcal{A}$ is often a subset of $\mathbb{R}^n$, where n is the dimensionality of the action space.

3.1.4 State Value

The agent's objective is to maximise its return, denoted as $G(\tau)$, over a trajectory τ. While some literature may confusingly R to represent return, just as the reward function, we adopt G as the standard notation. The return is often calculated as the discounted return up to the horizon, which includes the discount factor:

$$G(\tau) = \sum_{i=0}^{H} \gamma^i r_i \tag{2}$$

The expected return that a certain policy achieves, starting from a certain state $s \in \mathcal{S}$ is called the state-value $V : \mathcal{S} \to \mathbb{R}$. Mathematically, it is defined as:

$$V^\pi(s) = \mathbb{E}_{\tau \sim Pr(\tau)} \left[G(\tau) \mid s_0 = s \right] \tag{3}$$

where $V^\pi(s)$ is the state-value function under policy π, $\mathbb{E}_{\tau \sim Pr(\tau)}$ denotes the expected value given that the agent follows policy π on trace τ.

3.1.5 State-action Value

The state-action value $Q^\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, also known as the action-value function, quantifies the expected return of taking an arbitrary action $a \in \mathcal{A}$ in a state $s \in \mathcal{S}$ and following policy π thereafter. It is defined as:

$$Q^\pi(s, a) = \mathbb{E}_{\tau \sim Pr(\tau)} \left[G(\tau) \mid s_0 = s, a_0 = a \right] \tag{4}$$

where $Q^\pi(s, a)$ represents the expected return when starting from state s, taking action a, and subsequently following policy π. The state-action value is often abbreviated to $Q(s, a)$ and is also called the Q-value.

3.1.6 The Bellman Equation

The Bellman equation is a fundamental equation in RL that recursively expresses the relationship between the value of a state s and the values of successor states s' (Bellman, 1966). For the state-value function, the Bellman Equation shows the relationship between the value of the current state and the value of the next states:

$$V^\pi(s) = \mathbb{E}_{a \sim \pi(s)} \left[\mathbb{E}_{s' \sim P(\cdot|s,a)} \left[R(s, a, s') + \gamma V^\pi(s') \right] \right] \tag{5}$$

Here, the state-value under policy π is expressed as the sum of the immediate reward and the discounted value of the successor state s'. Because the successor state is stochastic, we take the expectation sampled from the transition function P and the expectation over actions sampled from policy π in state s.

The Bellman equation for the action-value function $Q^\pi(s, a)$ is given by the recursive relationship of the successor state s' and action a':

$$Q^\pi(s, a) = \mathbb{E}_{s' \sim P(\cdot|s,a)} \left[R(s, a, s') + \gamma \mathbb{E}_{a' \sim \pi(s')} [Q^\pi(s', a')] \right] \tag{6}$$

Here, the action-value under policy π is expressed as the sum of the immediate reward $R(s, a, s')$ and the discounted action-value of the successor state s', given action a' which is sampled from policy π in the expected successor state s'.

These recursive Bellman equations are used for RL algorithms to calculate (among others) value and policy functions recursively. These form the basis for iterative RL methods such as value iteration and policy iteration (Bellman, 1966), which in turn underpin more advanced value iteration techniques like Q-learning (Watkins and Dayan, 1992) and deep Q-networks (DQN) (Mnih et al., 2015), as well as advanced policy iteration techniques like REINFORCE (Williams, 1992) and Trust Region Policy Optimisation (TRPO) (Schulman et al., 2015).

3.1.7 Formal Definition of the RL Optimisation Problem

The RL optimisation problem aims to find an optimal policy π^* that maximises the expected return, defined as the cumulative reward received over time. This return, denoted as $J(\pi)$, is the expected sum of rewards obtained when following policy π:

$$J(\pi) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t r_t\right] \tag{7}$$

The RL optimisation problem is formally defined as finding the policy π^* that maximises $J(\pi)$:

$$\pi^* = \arg\max_{\pi} J(\pi) \tag{8}$$

3.1.8 Deep Reinforcement Learning

In Deep Reinforcement Learning (DRL), which we will just call RL in the context of this work, the state-value function and/or policy function are computable functions which depends on a set of parameters. In the deep RL settings, these parameters are the weight and biases of a neural network. A parameterised policy is often denoted as π_θ and a parameterised value estimator is denoted as $Q_\phi(s, a)$.

3.2 Formal Definition of the CityLearn Problem as an RL Problem

The CityLearn environment can be utilised in two different settings, the central agent and multi-agent setting. We express the central agent CityLearn problem as a discrete-time finite-horizon MDP, outlined in Section 3.1.1, where the transition function represents the dynamics of the CityLearn environment. The objective is to learn an optimal policy π that maximises the expected return.

In the multi-agent setting, the CityLearn problem can be expressed as a multi-agent Markov game (Littman, 1994), defined by $(\mathcal{S}, \mathcal{A}, T, R, \gamma, H)$. Here, $\mathcal{S}$ denotes the state space, $\mathcal{A}$ represents the shared action space. Each agent i has a local observation $o_i \subseteq s \in \mathcal{S}$. $P(s'|s, A)$ indicates the transition probability from state s to s' given the joint action $A = (a_1, \ldots, a_n)$ for all n agents, $R(s, A)$ denotes the shared reward function, and γ is the discount factor. Agents use policy $\pi_{\theta_i}(a_i|o_i)$ to select an action a_i based on their local observation o_i. The objective is to jointly optimise the discounted accumulated reward.

4 The CityLearn Environment

CityLearn is a Farama Foundation Gymnasium environment (Towers et al., 2022) designed to simulate a district of buildings with varying loads, electrical devices, and energy storage systems. CityLearn supports the implementation of various control strategies, such as rule-based systems, model-predictive systems, and single- and multi-agent RL algorithms to facilitate DR by leveraging energy storage systems for optimal charging and discharging cycles (Vazquez-Canteli et al., 2012) (Nweye et al., 2023a). Figure 2 illustrates the CityLearn environment. Each building in the environment may be equipped differently, with varying heating, cooling, and energy demand needs.

One of the key advantages of CityLearn is that it is a self-contained simulator which operates using pre-existing models. It can use input files sourced from energy simulators like EnergyPlus (Crawley et al., 2000) or Modelica (Mattsson et al., 1998) models, as well as from real-world data. CityLearn incorporates building energy models with hourly energy usage data and the energy output from PV arrays under varying weather conditions.

CityLearn is extensively customisable. Depending on the chosen dataset, the number of buildings in a district can be modified. Additionally, the available storage systems for each building can be adjusted, along with the sizes of distributed energy resources (DERs) such as photovoltaic (PV) arrays and other renewable energy sources. The availability and sizes of electrical storage systems (ESSs), thermal energy storage tanks, and heat pump capacities can also be modified. This allows for the design of various optimisation challenges, such as optimising for electricity price, carbon emissions, or load shaping of the

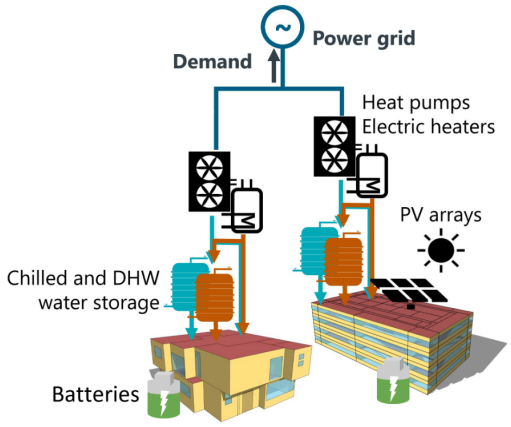


Figure 2: Visual description of the CityLearn environment. Each building in the district can be equipped with various kinds and sizes of energy storage systems and PV arrays which can be controlled by an agent to facilitate DR to the power grid. Blue and red indicate chilled and hot water storage, respectively. These storages can be filled by heaters or coolers and used for thermal energy needs. Energy from both the power grid and PV arrays can be utilised meet demand. Batteries can store PV-generated energy. Effective scheduling of battery power usage and stored heat or cooling can be applied for DR. Adapted from (Vazquez-Canteli et al., 2012).

grid electricity network. The framework supports both single-agent and multi-agent settings, providing a single reward for the former and a list of rewards, one for each building/agent, for the latter. The reward function in the CityLearn environment is customisable and can be adjusted to specific optimisation tasks.

4.1 Observations

CityLearn utilises datasets to define the simulation environment and provide observation values. These datasets include time series data for observations not dependent on agent actions, such as outdoor humidity and solar activity. Each building has a separate data file containing building-specific observations. Table 1 provides an example of the format of this data file.

Month	Hour	Day Type	...	Indoor Temp. [C]	Indoor Humidity [%]	Equip. Power [kWh]	...
1	1	1	...	22.74	83.26	9.44	...
1	2	1	...	22.69	83.38	9.68	...
1	3	1	...	22.69	83.29	8.48	...
1	4	1	...	22.70	83.23	7.85	...
1	5	1	...	22.73	82.97	7.76	...

Table 1: Example excerpt of a dataset used by CityLearn showing observations per hourly time step for a specific building. 'Month' and 'hour' denote the current time step, 'day type' denotes the type of the current day (e.g., weekday or holiday). 'Indoor temp.' and 'humidity' measure the temperature and humidity inside the building, respectively. 'Equipment power' indicates the amount of electrical power used by equipment during each time step.

The parameters "Month", "Hour", and "Day Type" denote the current date for building x, where "Day Type" can be 1 to 7 to indicate the day of the week, or could be assigned the value 8 to indicate a holiday. Additional data includes end-use loads, occupancy, solar generation, and time series data for indoor environmental variables. These variables are typically derived from simulations using energy modelling software, such as the EnergyPlus model (Crawley et al., 2000). The complete observation space details can be found in Appendix A.1. Additional variables, such as outdoor weather variables time series, carbon intensity rate time series and pricing data are also used in the simulation. These variables are shared between all buildings.

In the process of setting up a CityLearn environment, we have the option to omit certain observations or to normalise them. Notably, in some related studies, only a limited subset of observations is used. For example, Kathirgamanathan et al. (Kathirgamanathan et al., 2020) employed a small subset of 9 observations to train their SAC agent, while Nweye et al. (Nweye et al., 2023c) utilised a set of 14 observations for their MERLIN controller, which is based on independent SAC controllers that use transfer learning to speed up training. The choice of observation set depends on the specific problem being addressed in CityLearn. Researchers can experiment with different observation sets to evaluate their impact on the learning process of RL algorithms.

4.2 Actions

Based on the data in our dataset, a building's actions depend on the available devices. CityLearn includes several predefined devices listed in Table 2.

Device	**Unit**
Cooling storage	kWh/kWh capacity
Heating storage	kWh/kWh capacity
DHW storage	kWh/kWh capacity
Electrical storage	kWh/kWh capacity
Cooling device	kW/kW nominal
Heating device	kW/kW nominal

Table 2: Predefined devices that can be used in the CityLearn environment.

The continuous action range spans from -1.0 to 1.0, determining the percentage of a storage device's capacity to be either charged or discharged. For example, an action of -0.3 would attempt to discharge 30% of the battery's maximum capacity. Note that during simulation, there exists a maximum charging or discharging rate. For instance, if we have a fully charged battery (electrical storage) and input an action of -1.0, the discharge will be constrained to the maximum discharge rate per hour, as specified in the dataset.

4.3 Key Performance Indicators

CityLearn uses a set of Key Performance Indicators (KPIs) to evaluate the performance of an agent (Nagy et al., 2021) (Intelligent Environments Lab & AIcrowd, 2023). Depending on the optimisation challenge at hand, the KPIs to be optimised can be set for specific goals. The KPIs are calculated as a score between 0 and 1, proportional to the baseline score, which is the score achieved when the simulation is run without any interference from a controller. A score below 1 indicates that the controller is effectively leveraging its control systems. The KPIs used in this study are:

Ramping: Ramping is defined as the change in energy consumption between consecutive time steps:

$$r = \sum_{t=0}^{n-1} |E_t - E_{t-1}| \tag{9}$$

where E_t denotes the total neighbourhood net electricity consumption (in kWh) at time t. Minimising this metric involves minimising the change in energy consumption between consecutive time frames.

1 - Load Factor: The load factor is defined as the average load (in kWh) divided by the peak load in a specific time period. A high load factor means that electricity usage of a device remains relatively constant. Minimising the 1 - load factor ensures a more even distribution of electricity consumption throughout the day, thereby mitigating strain on electricity infrastructure during peak demand periods.

$$l = \left(\sum_{d=0}^{\frac{n}{h}} 1 - \frac{\left(\sum_{t=d \cdot h}^{d \cdot h + h - 1} E_t \right) \div h}{\max(E_{d \cdot h}, \ldots, E_{d \cdot h + h - 1})} \right) \div \frac{n}{h} \tag{10}$$

Here, n denotes the total number of time steps during the simulation, while h represents the number of hours per day, and d denotes the day index. The term $\frac{n}{h}$ corresponds to the number of days in the simulation, and $d \cdot h + h - 1$ signifies the end time step of a day. The numerator of the fraction calculates the average usage, while the denominator calculates the peak load.

Average Daily Peak: The net daily peak demand is the total energy consumption (in kWh) at a specific moment subtracted by the energy supplied from PV or energy storage systems. It represents the highest energy draw from the grid at a specific point in the day. The average daily peak is calculated as the average of each daily peak.

$$p_d = \left(\sum_{d=0}^{\frac{n}{h}} \max(E_{d \cdot h}, \ldots, E_{d \cdot h + h - 1}) \right) \div \frac{n}{h} \tag{11}$$

Peak Demand: The maximum value of electricity demand (in kWh) reached at any time step.

$$p_n = \max(E_0, \ldots, E_n) \tag{12}$$

Net Electricity Consumption: The total amount of electricity (in kWh) consumed.

$$E = \sum_{n=0}^{n} (E_0, \ldots, E_n) \tag{13}$$

Carbon Emissions: Total amount of carbon emissions (in kg). To convert this to a score dependent on the baseline, we calculate

$$G = \sum_{i=0}^{b-1} g^i_{\text{control}} \div \sum_{i=0}^{b-1} g^i_{\text{baseline}} \tag{14}$$

where the total number of buildings is indicated with b and $g = \sum_{i=0}^{n-1} \max(0, e_t \cdot B_t)$. Here n denotes the total number of time steps, e_t denotes the energy consumption at time step t and B_t denotes the carbon intensity or emission rate (in kgCO2e/kWh).

In addition to the mentioned KPIs, CityLearn includes various other KPIs that can be used to evaluate the performance of agents in different problem settings. In our work focusing on the DR problem, we evaluate our agents at test time based on the average score across all the aforementioned KPIs. Since CityLearn is an environment that can be used to address different challenges apart from the DR problem, these challenges can be evaluated with different KPIs (e.g., price cost for a pricing agent). Due to the adaptability of CityLearn, researchers can define their own reward function tailored to the specific problem at hand.

4.4 The CityLearn Challenge

Since the release of the CityLearn environment in 2020, the CityLearn Challenge (Vázquez-Canteli et al., 2020) (Nagy et al., 2021) (Nweye et al., 2022b) (Intelligent Environments Lab & AIcrowd, 2023) is hosted annually. Each challenge contains a different problem, with two of the challenges, namely the 2022 and 2023 editions hosted at NeurIPS. The CityLearn Challenge is a global competition that tasks participants with creating AI controller agents for the energy management of a microgrid. Contestants of each challenge are tasked with creating their own control agents along with their own custom reward function.

The CityLearn Challenge 2020

For the inaugural CityLearn Challenge in 2020, the dataset encompassed four sets of building data from distinct climate zones. Each climate featured data from 9 buildings, some equipped with PV arrays while others relied on obtaining energy from the grid. All buildings had access to a heat pump, electric water heater, chilled water tank, and a DHW (Domestic Hot Water) storage tank. They could undertake two actions: adjusting cooling storage and/or DHW storage. The objective was for agents to control these storage elements on an hourly basis over a simulated period of 4 years. Evaluation was conducted across various climate zones using pre-trained agents to assess agent resilience to changes in data. The evaluation conditions for this challenge were peak demand Equation (12), average daily peak Equation (11), ramping Equation (9), 1 - load factor Equation (10) and net electricity consumption Equation (13).

The winning solution had an average score of 0.879 across all evaluation metrics, giving an improvement over the RBC baseline of approximately 12.1%. The second place used a centralised Soft Actor Critic RL agent to achieve an average score of 0.967 (Kathirgamanathan et al., 2020). In other work, Chen et. al use a combination of Evolutionary Algorithms and MPC to achieve an average score of 0.832 or 16.8% (Chen et al., 2020).

The CityLearn Challenge 2021

For the second CityLearn challenge hosted in 2021, the dataset comprised 9 buildings with 4 years of simulated data. Unlike the previous challenge, buildings were now equipped with electrical storage in addition to DHW and chilled water storage, expanding the action space. The challenge was that the control policy could only be learned from a single episode of 4 years of data. Additionally, a new evaluation metric, carbon emissions, was introduced.

The evaluation conditions for this challenge were the same as those from the 2020 challenge Section 4.4 but also included carbon emissions Equation (14). The winners of the CityLearn Challenge 2021 used an evolutionary algorithm (EA) to achieve an average score of 0.962 across all evaluation metrics. This score was primarily influenced by the low ramping costs (Khattar and Jin, 2022).

The CityLearn Challenge 2022

For the third CityLearn challenge, the focus is only on efficiently using battery storage to flatten the energy demand curve from buildings. The dataset is a digital replica of EPRI's net-zero demonstrator in Fontana, CA (Nweye et al., 2022b) which consisted of the real-life end-use data of 17 buildings during 1 year. These buildings were studied for grid integration in zero net energy communities under the California Solar Initiative program, specifically examining buildings equipped with a PV array and a battery (Narayanamurthy et al., 2016). The only two evaluation KPIs used were net electricity consumption Equation (13) and carbon emissions Equation (14).

The winning solution employed an ensemble approach, combining gradient-boosted decision trees (GBDT) and the linear least squares model for load and solar generation forecasting. These predictions were then fed into a multi-agent Proximal Policy Optimisation (MAPPO) model. It achieved an average score of 0.728 across four metrics: emission cost (0.808), price cost (0.639), and grid cost (0.783) (Intelligent Environments Lab & AIcrowd, 2022). Furthermore, the PPO algorithm from Stable-Baselines3 achieved a score of 0.954 (Forbu, 2022).

The CityLearn Challenge 2023

The challenge utilises the dataset "End-Use Load Profiles for the U.S. Building Stock" for simulating 6 buildings during the course of 1 year. Initially, agents could only control 3 out of the 6 buildings. The objective was to assess whether the policies developed for these buildings could be generalised to control all 6 buildings for online evaluation. The challenge was structured to ensure that the agents' policies could adapt to unseen variations in building thermal dynamics and energy usage. Each building is equipped with a heat pump, electric heater, DHW storage, battery, and PV systems. The controllable devices include the cooling device, battery charge, and DHW storage. Unlike the 2022 challenge, the

PV systems in this scenario are sized for only 20% of the annual load, making zero-net energy unattainable.

The CityLearn Challenge 2023 uses a weighted score to evaluate agents based the same KPIs as the 2021 challenge, but also introduced additional KPIs such as all-time peak, 1 - thermal resilience and normalised unserved energy. At the time of writing, the CityLearn Challenge 2023 is still ongoing.

Summary

Four CityLearn challenges have been conducted to date, each presenting both real-life and simulated datasets. The challenges introduced different numbers of buildings, actions, simulation lengths and evaluation criteria as discussed in this section and are summarised in Table 3. Except for the 2021 challenge, for which the best agent used an EA, the most promising agents utilised RL techniques.

Challenge	Buildings	Actions	Simulation Length
2020	9	2	4 years
2021	9	3	4 years
2022	17	1	1 year
2023	6	3	1 year

Table 3: Summary of the past CityLearn Challenges. Each challenge featured varying district sizes, action spaces, and simulation lengths.

All of these challenges come with their own unique provided datasets. In this study, we aim to benchmark RL techniques of varying complexity. The dataset used for the CityLearn Challenge 2023 was created using synthetic building data. Building upon this, Nweye et al. (Nweye et al., 2023b) released a dataset containing the end-use data of 100 buildings in three different climate zones. In this study, we will utilise this dataset to compare single-agent RL approaches to multi-agent RL approaches on simulations of varying complexity, achieved by scaling the number of buildings in each problem. Unlike the datasets used in the CityLearn challenges, each of which contains a relatively small number of controllable buildings (as indicated in Table 3), the dataset used in this study contains considerably more buildings to test the scaling ability of different approaches. We will use the five KPIs introduced in Section 4.3 to measure the performance of the benchmarked agents. We will introduce the dataset used for this study in Section 5.1. The environment settings that will be used in this research will be detailed in Section 5.2.

5 Problem setting and dataset selection

In this research, we utilise a dataset similar to those introduced by the CityLearn challenges, which are specifically tailored to work with a Farama Foundation Gymnasium environment for benchmarking RL building control algorithms. Leveraging EnergyPlus physics-based simulation models, synthetic representations of buildings were created to emulate diverse real-world scenarios. These simulations incorporated factors such as diverse building types, sizes, occupancy patterns, and equipment configurations to make sure the dataset resembles real-world situations. Each time step in the dataset comprises one hour of simulated data.

5.1 Dataset

Unlike the datasets from the CityLearn challenges, which typically simulate a smaller number of buildings as discussed in Section 4.4, this dataset contains simulations for 100 buildings across three distinct climate zones (Nweye et al., 2023b). This expanded dataset enables us to benchmark RL algorithms on a larger scale, ranging from a hamlet of 5 buildings an entire district comprising 100 buildings. It also enables us to set the amount of actions, from only using the battery as controllable action or choosing both the battery and DHW storages as controllable actions. Since the scalability of centralised RL methods to such a large number of buildings, given their huge combined observation space, remains unclear (Kathirgamanathan et al., 2020), this dataset can be utilised to gain insights into this aspect.

Building Data

While the building data is synthetic. The end-use load profiles are based on the *End Use Load Profiles for the U.S. Building Stock* database by the National Renewable Energy Lab (NREL) (Wilson et al., 2022) (Wilson, 2017). They collect data from real-life sources utilities, smart bodes and publicly available data to generate 900.000 energy models for different buildings from residential buildings to commercial buildings. The data from this database was then used to generate a synthetic neighbourhood consisting of 100 residential buildings (Nweye et al., 2023b).

Each building comes with its own synthetically generated daily load and temperature profiles. Figure 3 shows an example of the load profiles of 3 buildings of our dataset. The building is identified by its specific building identifier which is shown on top of each figure.

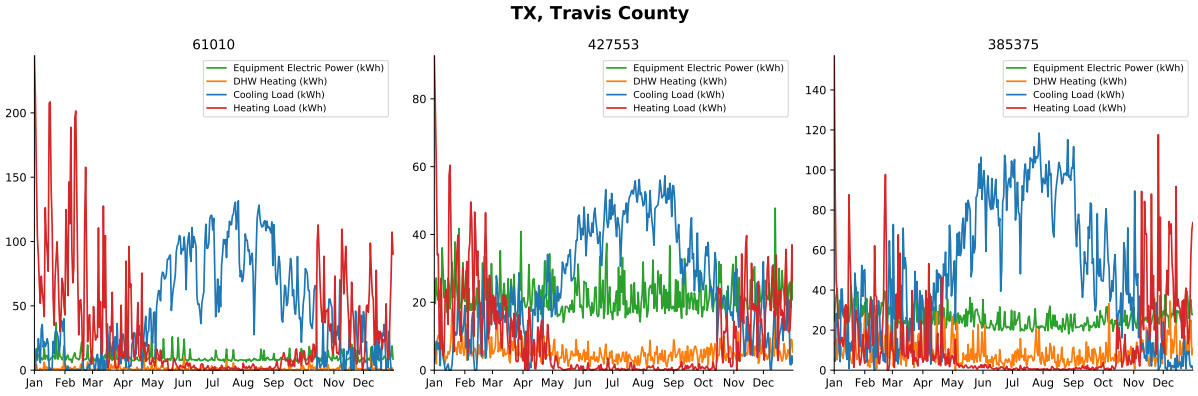


Figure 3: Example of the daily load profiles of 3 buildings in the dataset, illustrating the unpredictable changes in demand patterns of specific buildings over a year.

The figures are generated by aggregating daily load data from each building, summing up energy consumption at each hour of the day from our dataset. The equipment electric power load represents the electricity consumed by electrical appliances within the building per day. The DHW Heating (Domestic Hot Water) load accounts for the energy used to heat water. The cooling load represents the energy consumed by air conditioning systems. The heating load represents the energy used by heating systems such as furnaces, boilers, or heat pumps.

The dataset also includes indoor temperature data. An example of the indoor temperatures for 3 buildings in our dataset is shown in Figure 4. The temperature graphs are created by aggregating the hourly indoor temperature data for each day. The figure shows the minimum, average, and maximum indoor temperatures for each day.

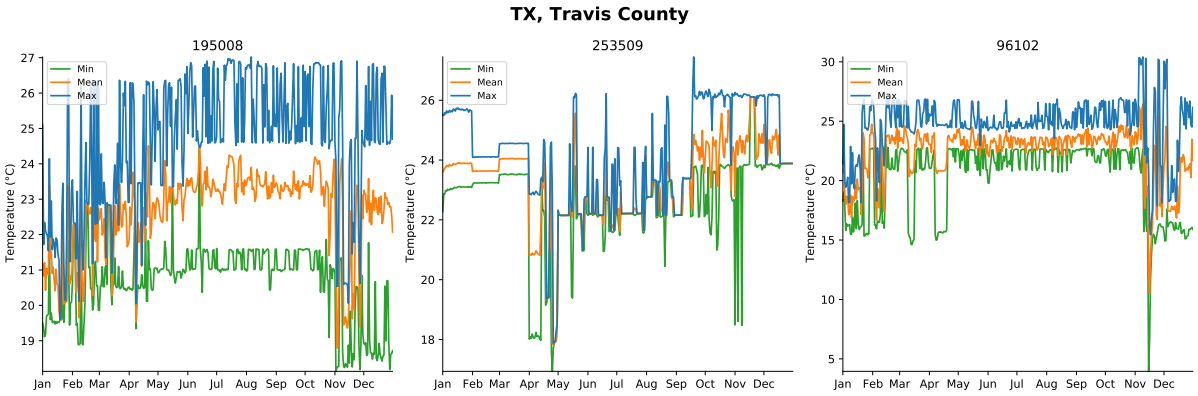


Figure 4: Example of the daily temperature profiles of 3 buildings in the dataset, showcasing the varying temperature patterns across different buildings.

Weather Data

The weather data comprises real-life meteorological year data from AMY (Wilson et al., 2022) and TMY3 data (Wilcox and Marion, 2008), covering the 2018 calendar year. This data is sourced from weather stations identified in the EULP database for three distinct climate zones: Almaeda Co., CA (marine climate), Travis Co., TX (hot-humid climate), and Chittenden Co., VT (cold climate). The average temperatures for each month in the Travix Co., TX climate zone are illustrated in Figure 5.

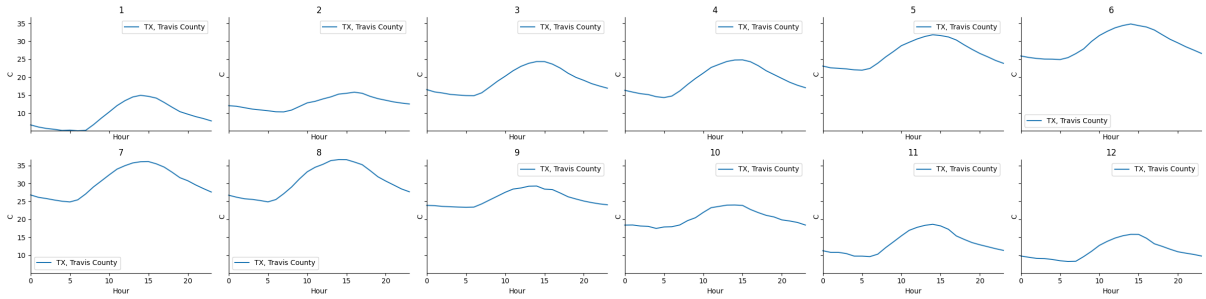


Figure 5: Monthly average daily temperature data for the Travix Co., TX climate zone, highlighting the difference in temperatures during the year.

Carbon Intensity Data

The carbon intensity data was derived from the Electric Reliability Council of Texas (ERCOT) grid fuel mix, representing the proportion of renewable energy generation to the total energy generated at specific time steps.

The average carbon intensity (g/kWh) of a region at any time is given by:

$$\text{Carbon Intensity}_{\text{avg}} = \frac{\sum(E_i \times \text{CEF}_i)}{\sum E_i}$$

where E_i represents the electricity generated (in MWh) by source i, and CEF_i denotes its carbon emission factor (g/kWh) (Maji et al., 2022).

Pricing Data

Electricity prices for each step of the dataset are sourced from the NREL database (Wilson, 2017). They include the current electricity pricing (in $\$/kWh$), as well as predictions for electricity pricing at 6-hour, 12-hour, and 23-hour intervals.

5.2 Benchmarking Environments

In this research, we aim to compare a set of state-of-the art RL algorithms on the continuous CityLearn demand response problem introducing scenarios with increasing dimensionality. To comprehend strengths and weaknesses of single-agent vs. multi-agent approaches, we will examine three scenarios. In the first scenario, we consider only 5 buildings. In the second scenario, we scale up to 10 buildings, in the third scenario, we expand to 50 buildings. This setup is chosen so RL algorithms can be compared to each other on different problem scales.

We will utilise the dataset elaborated in Section 5.1 to simulate the environment for the various scenarios outlined in Section 6. Figure 6 shows an overview of the systems and interactions between the buildings contained in the dataset and the grid in the multi-agent scenario. According to the end-use load and energy State of Charge (SoC) the agent can plan to charge or discharge an energy storing device.
In a single-agent RL scenario, a central agent oversees the energy management of all buildings in the neighbourhood, including controlling both the battery and DHW (Domestic Hot Water) storage, determining the amount of energy to store or release at each time step. In a multi-agent setting, each building has its own control agent which controls the battery and DHW storage for that building. A

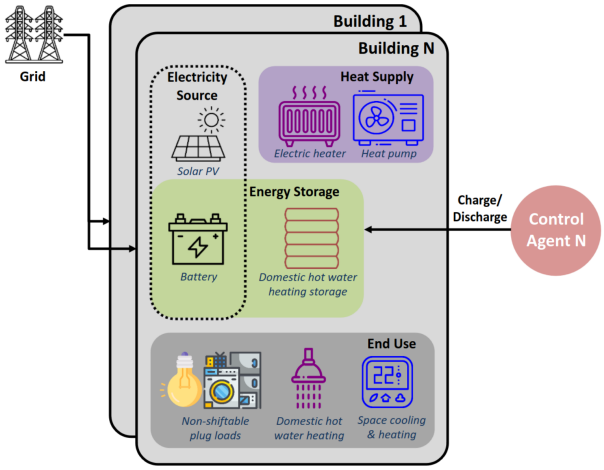


Figure 6: Interaction of the control agent with the CityLearn environment in a multi-agent scenario. Each agent has access to its own dataset and can control the battery and DHW energy storages to facilitate DR. Adapted from (Nweye et al., 2023a).

simulation in our environment takes 1464 time steps, or 61 days, covering June and July of the training data described in Section 5.1.

Actions

The action space for the scenario where the agent controls both energy storages is two-dimensional, ranging from -1.0 to 1.0. Both actions determine the percentage of the maximum storage to be charged or discharged.

Observations

The agent's observation space consists of various components providing information about the environment. Table 4 summaries these components.

Observation	Description
District-level temporal data	- Day of the week - Hour of the day
Weather observations	- Dry-bulb temperature - Direct solar irradiance - Six-hour forecast for temperature and solar irradiance - Twelve-hour forecast for temperature and solar irradiance
Building-specific states	- Solar generation - Battery state of charge (SoC) - Domestic hot water (DHW) storage state of charge (SoC) - Plug loads - Net loads - Net electricity consumption

Table 4: Observation space for our benchmarking environment. Agents use district level temporal data, weather observations and building-specific states to navigate the simulation.

These observations are preprocessed using cyclical transformation, one-hot encoding, and min-max normalisation to facilitate the learning process. Appendix A.1 shows a detailed table with the complete CityLearn observation space, which observations are shared or building-specific and which observations are used by the RL agents in this study.

Reward Function

The reward function aims to minimise net electricity consumption E at each time step while encouraging net-zero energy usage. This means the agent strives to maximise its use of self-generated PV energy and minimise reliance on energy from the grid. The penalty term p dynamically adjusts based on the state of charge (SoC) of the battery and DHW storage, as outlined in Equation (16). This reward function is included with the dataset as the default.

$$r = -p \cdot |E| \tag{15}$$

$$p = 2 + |E| \cdot (\text{SoC}_{\text{Battery}} + \text{SoC}_{\text{DHW storage}}) \tag{16}$$

When the agent exports surplus energy to the grid (i.e., $E < 0$), it is penalised less if the energy storages (battery and DHW storage) are charged to capacity. This encourages the agent to maximise self-consumption of generated energy rather than exporting it. Conversely, if the agent imports energy from the grid (i.e., $E > 0$) and the energy storages are near capacity, a penalty is imposed. This discourages unnecessary grid energy usage when storage capacities are sufficient.

Evaluation

After training the RL agents on the training dataset, we evaluate the performances of the agents on a test set covering one month of training data from August. At the end of the test simulation, we evaluate the agents based on 5 KPIs, namely net electricity consumption Equation (13), average daily peak Equation (11), ramping Equation (9), peak demand Equation (12) and 1 - load factor Equation (10).

6 Methods

In this section, we will introduce the benchmarked RL algorithms used in both central agent and individual multi-agent settings. Additionally, we will discuss a central training, decentralised execution (CTDE) method that has shown promise in other multi-agent settings. For each setting, we will benchmark state-of-the-art RL methods for continuous control problems like CityLearn. First, we will explain the classes of RL algorithms in Section 6.1, Section 6.2 and Section 6.3 necessary to understand the algorithms used in this work. Next, we will describe the benchmarked algorithms and their comparative differences in Section 6.4. Following this, we will outline the experimental setup in Section 6.5. Finally, in Section 7, we will discuss the performance of the described benchmarked RL methods and compare them against each other and the RBC baseline across the three settings of increasing complexity.

RL algorithms can be divided into three classes. The first class consists of value-based methods, where an agent learns a value function $Q_\phi(s, a)$ that estimates the expected return of being in a particular state and taking a particular action. The second class comprises policy-based methods, which directly learn a policy π_θ without estimating the value function. The third class of RL algorithms encompasses actor-critic methods, which combine elements of both value-based and policy-based approaches. In these methods, an agent learns a policy π_θ using an estimated value function.

As laid out in Section 4, the CityLearn environment supports both single-agent and multi-agent settings. In the single-agent setting, one agent is presented with the complete observation space of all global district level observations and building-level observations. It then returns a list of actions for each building. In the multi-agent setting, each building is controlled by its own agent. The environment gives the building-specific observations and district level observations to each agent, which returns an action per controllable device, as described in Section 4.2.

Intuitively, the single-agent approach should perform better on this task, as it has a complete description of the system in the combined observations of all buildings and it should be able to learn a policy that ensures coordination between the building energy demand profiles to smooth district-level energy demand peaks. The multi-agent approach could be more practical in a real-world setting since the single-agent approach requires connecting the controller to all buildings, which presents a significant bottleneck. The multi-agent approach allows each agent to operate independently from other agents, which should also

scale better in scenarios with larger districts.

In CTDE methods, we train a set of agent by sharing information between them during training to learn individual policies. During test time, only these trained policies are used to generate the actions of the agent, which means we can implement this class of algorithms in a decentralised way in the real world. CTDE methods are mostly applied using actor-critic methods which learn a centralised critic which takes a global or joint observation as input (Lowe et al., 2017) (Rashid et al., 2020). This means we train a shared value function during training, which serves as the value estimator for the policy function. During deployment, only the policy function is used.

The most straightforward approach of implementing a multi-agent algorithm is decentralised learning, where each agent optimises its own reward independently. In theory, agents are still able to learn from the actions of other agents due to the global state, which is a function of all agents' actions.

In short, we compare three classes of approaches: Central agent approaches, where agents are both trained and executed in a centralised fashion (CTCE). Independent agent approaches, where agents are trained independently and also executed independently (DEDE); and a hybrid approach, where agents are trained centrally but executed independently (CTDE). In Section 6.4, we will introduce the benchmarked algorithms that will be evaluated both in a central agent setting and in an independent multi-agent setting. These algorithms represent state-of-the-art techniques in model-free RL for continuous control. Additionally, in Section 7.4, we will present the benchmarked state-of-the-art CTDE method for comparison.

6.1 Value-based Methods

Value-based reinforcement learning algorithms aim to learn the value function associated with each state-action pair. The value function, denoted as $Q(s, a)$, estimates the expected return the agent can achieve by taking a specific action a in a given state s and following a specific policy thereafter. Well-known value based algorithms are Q-learning (Watkins and Dayan, 1992) and Deep Q-Networks (DQN) (Mnih et al., 2015), which iteratively update the value function based on observed experiences.

In discrete environments, finding the action that maximises the estimated value function often involves selecting the action with the highest value among a finite set of possible actions. This can be represented as: $a^* = \arg\max_{a \in \mathcal{A}} Q(s, a)$.

In continuous environments like CityLearn, where the action space is continuous, traditional value-based methods might face challenges. The number of possible values for the computation of arg max becomes infinite in this case. One solution to this problem would be to discretise the action space into bins, transforming the continuous action space into a discrete one. By doing so, the problem becomes learnable to traditional value-based algorithms, but we will lose freedom of movement by doing this. The more bins we choose, the more freedom of movement or action we get, but the higher the computational demand.

6.2 Policy-based Methods

Policy-based methods directly learn continuous or stochastic policies without estimating the value function. Instead of learning a value function $Q(s, a)$, policy-based methods optimise the policy $\pi_\theta(a|s)$ directly by adjusting the parameters θ in the direction of the performance gradient $\nabla_\theta J(\pi_\theta)$ to maximise the expected return.

The foundation of policy-based algorithms lies in the policy gradient theorem which states that policy iteration with arbitrary differentiable function approximation is convergent to a locally optimal policy (Sutton et al., 1999):

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{s \sim \rho_\pi, a \sim \pi_\theta}[\nabla_\theta \log \pi_\theta(a|s) Q^\pi(s, a)] \tag{17}$$

One common policy-gradient approach is Williams's episodic REINFORCE algorithm (Williams, 1992), which uses the policy gradient theorem to update the policy parameters. The update rule for REINFORCE

is given by: $\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(a|s)\hat{G}$ where α is the learning rate, and $\hat{G}$ is an estimation of $Q^\pi(s,a)$. One way of estimating this value suggested by William is using a sample return r_t^γ. This approach allows for direct optimisation of the policy without estimating a value function and thus without calculating $\arg\max$, making it suitable for continuous action spaces.

6.3 Actor-Critic Methods

Actor-critic methods combine value-based and policy-based approaches into one. With classical policy-based methods such as REINFORCE, trajectories are generated using random actions to estimate the action-value function $Q^\pi(s,a)$, which causes high variance in the policy gradient estimates which causes learning to go slowly (Sutton and Barto, 2018).

Actor-critic methods address this by using a value function (the critic) to guide the policy update (the actor). The critic evaluates the action taken by the actor by estimating the value function $Q(s,a)$ or the advantage function $A(s,a)$. The actor updates the policy parameters θ using the gradients provided by the critic. If the value function estimation is sufficiently good, this reduces the variance of the gradient estimates and leads to more stable and efficient learning. However, a potential downside is that if the value function learned by the critic is inaccurate, the policy is optimised based on incorrect evaluations. This can lead to sub-optimal performance or even divergence. In practice, actor-critic methods outperform policy-based and value-based methods on several popular benchmarks (Konda and Tsitsiklis, 1999). This is why all of our chosen benchmarked algorithms are actor-critic methods.

6.4 Benchmarked Algorithms

The algorithms we will benchmark are Deep Deterministic Policy Gradient (DDPG) (Lillicrap et al., 2015), Twin Delayed Deep Deterministic Policy Gradient (TD3) (Fujimoto et al., 2018), Proximal Policy Optimisation (PPO) (Schulman et al., 2017) and Soft Actor-Critic (SAC) (Haarnoja et al., 2018). We will run these on every problem setting of increasing difficulty in both a central agent setting and an independent multi-agent setting. Additionally, we will compare these approaches to the multi-agent PPO (MAPPO) (Yu et al., 2022) algorithm, which is used in a CTDE setting.

6.4.1 Deep Deterministic Policy Gradient

Deep Deterministic Policy Gradient (DDPG) is an actor-critic reinforcement learning algorithm that directly learns a deterministic policy using an estimated state-value function (Lillicrap et al., 2015). It builds upon the Deterministic Policy Gradient (DPG) algorithm (Silver et al., 2014), extending its capabilities to handle deterministic action spaces to deep RL. The 'deep' aspect of deep RL refers to use of deep neural networks (DNNs) to estimate both policy and state-value functions. DDPG employs a DNN as the actor to approximate the deterministic policy function and update policy parameters via the DPG theorem, while a separate DNN (the critic), estimates the state-value function. Typical actor-critic methods model a stochastic policy as a probability distribution over actions, while DDPG focuses on learning a deterministic policy that maps states directly to specific actions.

DDPG utilises experience replay and target networks to enhance stability during training. Experience replay entails storing transitions (s_t, a_t, r_t, s_{t+1}) in a replay buffer $(\mathcal{D})$ and randomly sampling mini-batches for training. This approach breaks the correlation between consecutive updates and enhances data efficiency. Target networks, which are delayed copies of the actor and critic networks updated less frequently, provide stable target values for the critic network. This causes the value estimation to diverge less, increasing training time due to the slower updating networks, but greatly increases training stability.

DDPG is designed to handle continuous action spaces. In these spaces, directly computing $\max_a Q^*(s,a)$ is infeasible due to the infinity of possible actions in a continuous action space. Instead, we use the policy $\mu_\theta(s)$ to directly output a single action instead of a probability distribution over actions, approximating $\max_a Q(s,a) \approx Q(s, \mu_\theta(s))$. The actor is updated by following the gradient of the expected return with respect to its parameters:

$$\nabla_\theta J \approx \mathbb{E}_{s\sim\mathcal{D}}[\nabla_\theta Q(s, \mu_\theta(s))] \tag{18}$$

The critic network estimates the Q-value of a state-action pair (s, a). It is trained to minimise the mean squared Bellman error (MSBE), which measures the discrepancy between the predicted Q-value and the target Q-value, indicating how closely Q_ϕ satisfies the Bellman equation.

$$L(\phi) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\left[(Q_\phi(s, a) - y)^2\right] \tag{19}$$

where the target y is given by:

$$y = r + \gamma Q_\phi(s', \mu_\theta(s')) \tag{20}$$

The critic updates its parameters ϕ by minimising this loss. Since states are sampled from the experience replay buffer, DDPG is considered an *off-policy* algorithm, meaning that it learns from states that were generated by a different policy than the one that is being optimised.

6.4.2 Twin Delayed Deep Deterministic Policy Gradient

Twin Delayed Deep Deterministic Policy Gradient (TD3) is an extension of DDPG that aims to further improve training stability and performance (Fujimoto et al., 2018). One of the key enhancements in TD3 is the introduction of twin critics, which are two separate Q-value estimators. This setup is similar to the trick used in the double Q-learning algorithm, which helps mitigate overestimation bias in the Q-value estimates (Van Hasselt et al., 2016). TD3 uses the two critics' outputs and uses the smaller of the two Q-values to form the targets in the loss functions during training to provide more robust Q-value estimates, reducing the likelihood of overestimation.

$$y = r + \gamma \min(Q_{\phi_1}(s', \mu_\theta(s')), Q_{\phi_2}(s', \mu_\theta(s'))) \tag{21}$$

Similar to DDPG, TD3 also employs target networks and experience replay to enhance training stability. However, TD3 introduces a technique called target policy smoothing to stabilise training further. Target policy smoothing adds noise to the target actions during training, which helps prevent overfitting to Q-values. The target action a' is defined as:

$$a'(s') = \mu_{\theta_{\text{target}}}(s') + \epsilon \tag{22}$$

where $\epsilon \sim \text{clip}(\mathcal{N}(0, \sigma), -c, c)$ is clipped noise sampled from a normal distribution with mean 0 and standard deviation σ, and clipped to the range $[-c, c]$. The target value y then becomes:

$$y = r + \gamma \min(Q_{\phi_1}(s', a'(s')), Q_{\phi_2}(s', a'(s'))) \tag{23}$$

6.4.3 Soft Actor-Critic

Soft Actor-Critic (SAC) is an off-policy actor-critic algorithm that incorporates the maximum entropy framework to improve exploration and stability during training (Haarnoja et al., 2018). SAC aims to maximise both the expected return and the entropy of the policy, promoting more stochastic policies that facilitate exploration. It was designed to handle continuous action spaces and aims to improve exploration and robustness of the learning process. The key idea in SAC is to augment the standard maximum reward objective with an entropy term, encouraging the agent to maximise rewards while acting as randomly as possible. Unlike DDPG, which outputs a single deterministic action, the SAC policy outputs a probability distribution P and the entropy of this distribution $\mathcal{H}$ measures the randomness of the action selection (Equation (24)). SAC shares a couple of tricks used in the TD3 algorithm, which was developed concurrently. Both algorithms use the MSBE error to learn the state-action values. SAC also uses target networks and the clipped double Q trick.

$$\mathcal{H}(P) = \mathbb{E}_{x\sim P}\left[-\log P(x)\right] \tag{24}$$

In the context of maximum entropy, the agent receives an additional reward that correlates with the policy's entropy during that specific time step. This transforms the RL problem into:

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty} \gamma^t (r_t + \alpha\mathcal{H}(\pi(\cdot|s_t))\right] \tag{25}$$

where α denotes the trade-off parameter between rewards and entropy. Due to the inherent exploration in the stochastic policy, SAC can explore in an on-policy way without sacrificing exploration, unlike DDPG where actions are deterministic. At test time, the stochasticity of the policy is removed, and the mean action is used instead of a sample from the distribution.

6.4.4 Proximal Policy Optimisation

Proximal Policy Optimization (PPO) is a policy gradient method designed to ensure more stable learning (Schulman et al., 2017). In traditional policy gradient methods, updating the gradient with the loss of the policy can leads to destructively large policy updates (Schulman et al., 2017). To address this issue and make learning more stable, Schulman et al. proposed Proximal Policy Optimization. PPO constrains the size of the policy update by introducing a clipped surrogate objective, which prevents large policy changes. This clipping ensures that the new policy stays close to the old policy, preventing policy oscillations and improving stability. The surrogate objective function is given by:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min\left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right)\right] \tag{26}$$

where θ denotes the parameters of the policy network, and $r_t(\theta)$ is the probability ratio between the new policy and the old policy, defined as:

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \tag{27}$$

Here, $\pi_\theta(a_t|s_t)$ represents the probability of taking action a_t given state s_t under the new policy parameterised by θ, and $\pi_{\theta_{\text{old}}}(a_t|s_t)$ represents the probability under the old policy. $\hat{A}_t$ is an estimate of the advantage function at time step t, which measures how good the action a_t is compared to the average action at state s_t. ϵ is a hyperparameter that controls the extent of the clipping. The clip function is applied to ensure that the probability ratio $r_t(\theta)$ is within a certain range, defined as $[1-\epsilon, 1+\epsilon]$. The clipped surrogate objective encourages the policy update to stay close to the previous policy, preventing large changes that could lead to instability.

6.4.5 Multi-agent PPO

Multi-agent Proximal Policy Optimisation (MAPPO) is an adaptation of the traditional PPO algorithm designed for multi-agent environments (Yu et al., 2022). Instead of modifying the algorithm, the authors implement PPO in a multi-agent setting by utilising a shared value function $V_\phi(s)$, which considers the observations of all agents. MAPPO introduces centralised training and decentralised execution (CTDE) for the PPO algorithm, enabling each agent to learn its own policy while sharing their value networks with other agents during training. This approach facilitates agents in learning coordination and cooperation strategies while preserving scalability and efficiency. Consequently, multi-agent PPO does not have restrictions at test time because the shared critic is not utilised; only the policy network, which was trained with the shared critic, is used.

6.5 Experimental Setup

In this section, we will include the full spectrum of different environments used, with different problem dimensionalities. We will use CityLearn version 1.8.0 and the environments detailed in Section 5.2 with a varying number of buildings (5, 10 and 50). In each of these problem settings, we will compare the central agent methods (CTCE) described in Section 6 against each other and the RBC baseline. Additionally, we compare independent agent methods (DTDE) under the same problem conditions. We will compare the best performing single-agent method with the best performing independent multi-agent method. Furthermore, we will compare the performance of both central and independent agent approaches against the dependent multi-agent (CTDE) algorithm MAPPO, described in Section 7.4.

6.5.1 Hyperparameters and network architectures

We will use the Stable Baselines3 library as the code base for our agents (Raffin et al., 2021). Each experiment will use the hyperparameters and network architectures detailed in Appendix A.2. The default hyperparameters in the Stable Baselines 3 library are chosen for their ability to generalise well across various environments (Raffin et al., 2021), while the network architectures are based on those described in the original papers that proposed each algorithm. The standard PPO network architecture features an actor and a critic network, each with two hidden layers of 64 neurons. The standard SAC and DDPG architectures comprise an actor and a critic network, both with two hidden layers of 256 neurons each. The TD3 architecture includes an actor and a critic network with two hidden layers of 400 neurons each, as well as target networks with two hidden layers of 300 neurons each.

6.5.2 Network Parameters

To accommodate the increased size of the observation space in larger problem dimensions, we expanded the neural networks of the central agents to three hidden layers with 1024 neurons each, compared to the standard sizes detailed in Appendix A.2. This configuration was determined through experimentation with various network sizes on the 50-building problem. This also keeps the benchmarking fair between the independent and central agents, as the combined number of parameters for the independent agents also scale with the problem size. Larger network sizes beyond this point did not yield better performance. Note that this change was only made for the central agents in the 50-building scenario. Theoretically, increasing network size could lead to overfitting, as more parameters could enable the network to memorise sequences rather than generalise. However, empirical results suggest that even with significantly larger networks, this is not observed (Zhang et al., 2021). The smaller network sizes for both the independent and central agents in the smaller district sizes could potentially lead to underfitting. We will delve into this topic further in Section 8.

6.5.3 Training Methodology

Each agent is trained until the loss plateaus and the reward curve stabilises. Notably, for central agent methods, we observe that performance improvement slows significantly after approximately 300 training episodes (439200 time steps), prompting us to stop training at this point. In contrast, independent agent methods tend to converge much faster, usually within 40 training episodes (58560 time steps). Each experiment was run using one seed, due to the high computational costs of the experiments.

To make the learning curves of central agents clearer, we smooth them using a running average calculated over 10 training episodes. For independent algorithms, we group individual training curves using their mean values, and we show the standard error of this grouping in the training graphs.

6.5.4 Train-test Split

We run each environment for 1464 time steps, or 61 days, covering June and July of the training data described in Section 5.1. We then test the agent's performances on the test set covering one month of training data from August. The selection of this training data is the same as that recommended by the authors of the environment (Nweye et al., 2023b). At test time, we will compare all algorithms against each other according to the first five KPI scores elaborated in Section 4.3.

6.5.5 Computational Resources

The central agent experiments were run on a local machine with a NVIDIA GeForce GTX 1660S GPU. The independent agent experiments were run on the ALICE high-performance computing (HPC) cluster provided by Leiden University with a Intel Xeon Gold 6126 2.6GHz 12-Core, using a many parallel processes as there are agents.

6.5.6 Baseline

All agents are compared to the RBC baseline. The RBC is provided with CityLearn version 1.8.0, and its policy is designed to manage the storage systems as follows: discharge 2.0% of its maximum capacity

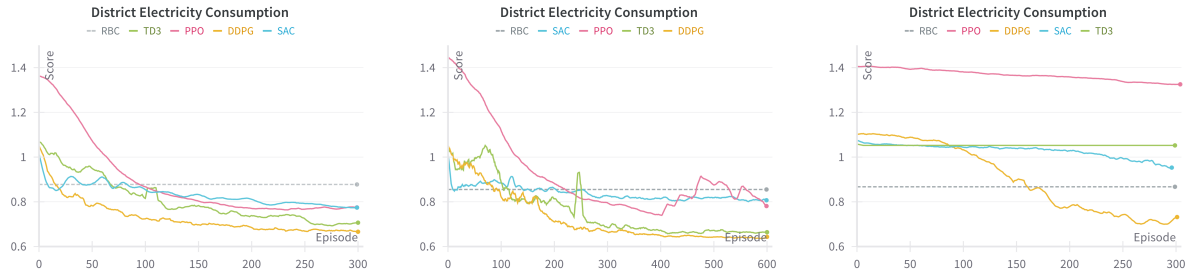
every hour between 7:00 AM and 3:00 PM, discharge 4.4% between 4:00 PM and 6:00 PM, discharge 2.4% between 7:00 PM and 10:00 PM, charge 3.4% between 11:00 PM and midnight, and charge 5.532% at every other hour. It seems that the RBC was originally designed using a different dataset, which caused its schedule to be misaligned with the actual day and night times in the Texas dataset. To correct this, we shifted the RBC's output by 15 hours so that its policy aligns properly with the day and night cycles in the Texas dataset.

7 Results

In this section, we present the results of our benchmarking study and evaluate the performance of central agent, independent multi-agent, and dependent multi-agent approaches. We begin with the results of the central agent approaches in Section 7.1, followed by the independent agent approaches in Section 7.3 and finally compare the performances of central agents and independent agents in Section 7.3. In Section 8, we will provide an in-depth discussion of the most significant results of this study.

7.1 Central Agent

For the central agent setting, we benchmarked four algorithms against the RBC baseline in our environment across three different district sizes. The agents' performance curves during training for these experiments are depicted in Figure 7.



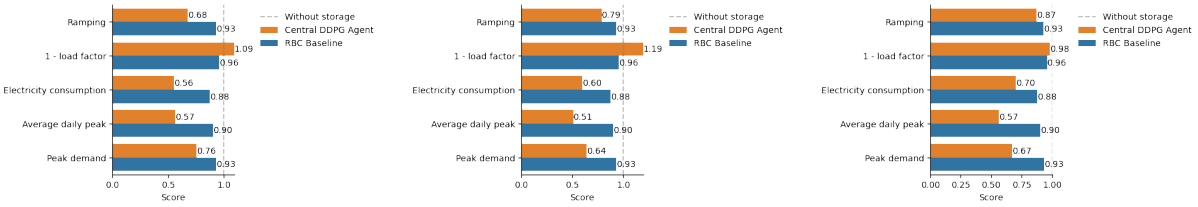
(a) District with 5 buildings **(b)** District with 10 buildings **(c)** District with 50 buildings

Figure 7: Performance curves for the benchmarked algorithms. All algorithms were trained until learning plateaus. For the 5- and 50 building settings, this was around 300 episodes. In the 10 building setting, the PPO algorithm stops improving after 400 time steps.

In the 5 building setting, the DDPG algorithm seem to perform best with a score of 0.73 averaged over all measured KPIs (explained in Section 4.3) during test time, as shown in Figure 8 and Table 5. Notably, all algorithms surpassed the RBC baseline in this scenario.

In the 10-building scenario, both DDPG and TD3 achieved similar scores. However, in the 50-building scenario, the PPO agent failed to solve the environment. The TD3 agent also cannot learn a policy that gets it below the 1 score evaluation, indicating a similar performance of a building with no battery or DHW heating, which can be seen as a local minimum solution to the problem. Only the DDPG agent successfully tackled this scenario, albeit with a lower score compared to the other settings.

In the 10-building scenario, both DDPG and TD3 score similarly to each other. In the 50-building scenario, the PPO agent is no longer able to solve the environment. Only the DDPG agent is able to solve this scenario outperforming the RBC, albeit with a lower score than the other two scenarios.



(a) District with 5 buildings **(b)** District with 10 buildings **(c)** District with 50 buildings

Figure 8: Neighbourhood-level KPI summary for the 5, 10 and 50 building CityLearn problems using the DDPG agent (which is the best performing agent in all environments) in the testing environment.

The snapshot of electricity consumption profiles by the DDPG agent on the 10 building problem reveals the trained policy achieves significantly lower average daily peaks compared to the consumption profile without storage (Figure 9). This reduction was consistent across all buildings, as indicated by the battery SoC averages shown in green.

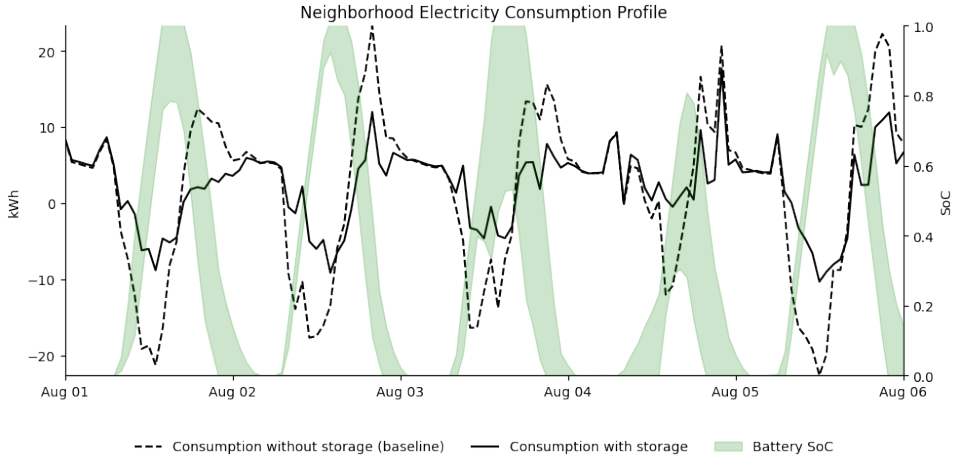
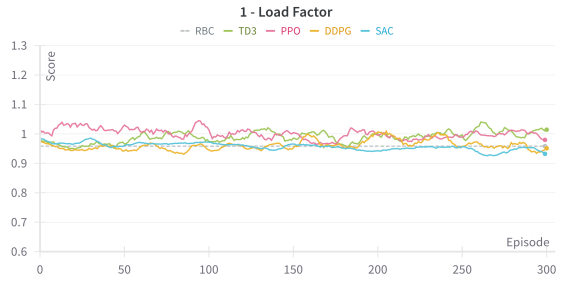
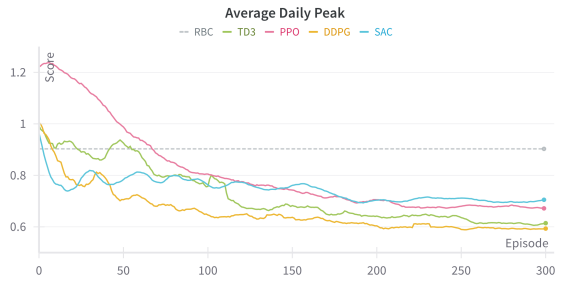


Figure 9: Snapshot of the electricity consumption and battery SoC of the DDPG agent on the 10-building CityLearn problem. The green band represents the aggregated data from all buildings in the environment, indicating that the agent learns to individually control different buildings. The black line shows the electricity consumption when efficiently utilising energy storage devices, controlled by the DDPG agent. This demonstrates the agent's ability to flatten the overall energy demand curve, showcasing its effectiveness in demand response.

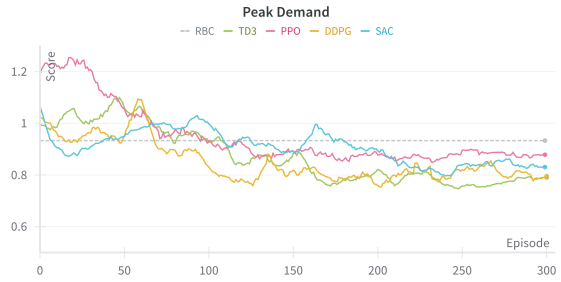
Training the central agents reveals that all algorithms are able to learn and decrease the 5 KPIs by a significant amount. While the reward function focuses on decreasing the net electricity consumption of the district, the policy also reduces ramping, peak demand, and the average daily peak. However, the 1-load factor does not seem to be significantly affected during training. The performance curves for those KPIs can be found in Figure 10. In the 5-building setting of the CityLearn problem, TD3 and DDPG outperformed the PPO and SAC algorithms.



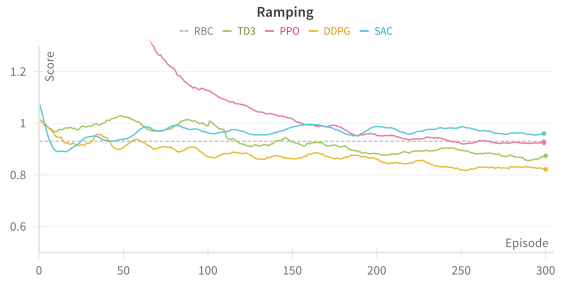
(a) 1 - load factor score during training for the DDPG agent.



(b) Average daily peak score during training for the DDPG agent.



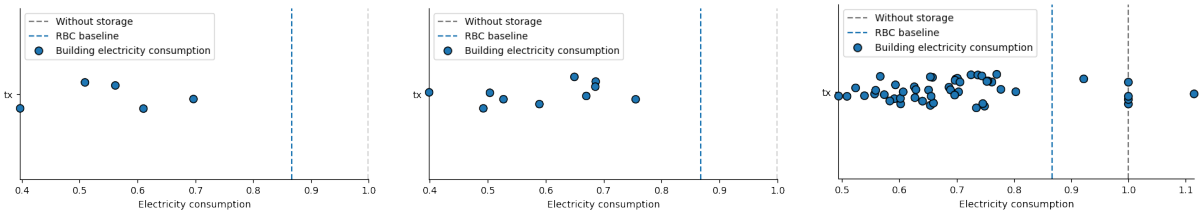
(c) Peak demand score during training for the DDPG agent.



(d) Ramping score during training for the DDPG agent.

Figure 10: KPI scores during training for the DDPG agent. It is evident that while the training objective is reducing electricity consumption, the trained policy also lowers average daily peak, peak demand and ramping KPIs.

The distribution of electricity KPI per building is visualised in Figure 11, illustrating the variability in performance across different buildings within each scenario. Note that the baseline score is averaged, but in reality, it varies for each building due to their different load profiles.



(a) District with 5 buildings

(b) District with 10 buildings

(c) District with 50 buildings

Figure 11: KPI distribution plot for the electricity consumption of every building in each scenario for the best performing central DDPG agent. Clearly, not every building is able to achieves the same score. This variation can be attributed to different load profiles, which allow some buildings to benefit more from energy storage devices. In the 50-building scenario, visualised in Figure 11c, it is evident that the central agent is unable to improve electricity consumption for four buildings, and for one building, the performance is worse than when not using energy storage.

The snapshot of the electricity consumption and battery SoC of the DDPG agent on the 50-building problem, depicted in Figure 12, shows that the agent clearly learns different policies for different buildings. The wider area of the plot signifies the different policies for different buildings. There is a clear pattern for the aggregated data of the 50 buildings which is not simply repeated each time, but seems to be tailored to the future consumption, as shown by closer charging and discharging times of the battery from August 4 to 5.

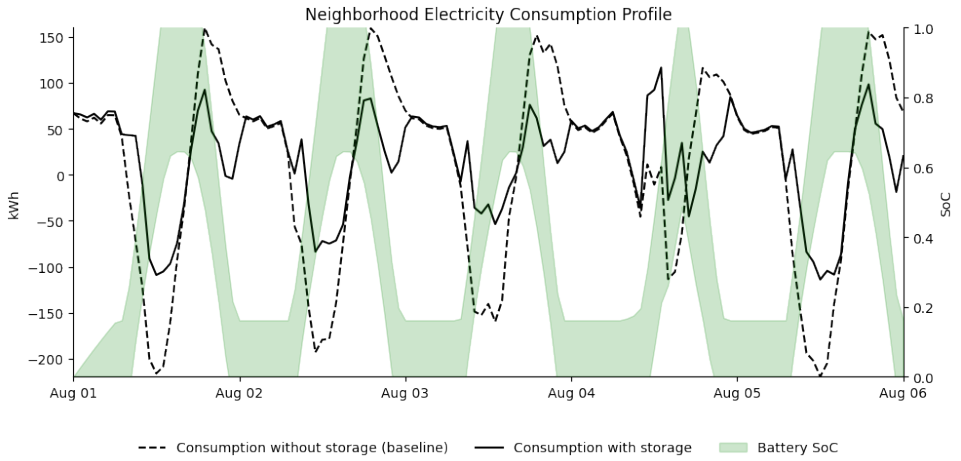
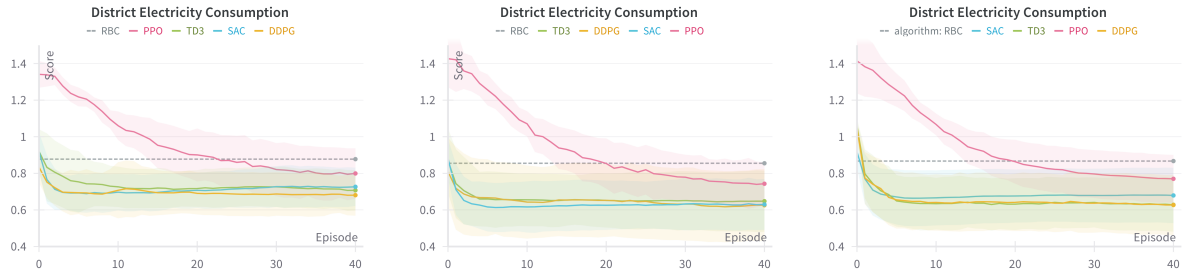


Figure 12: Snapshot of the electricity consumption profile for the DDPG agent on a segment of the test set spanning from August 1 to August 6. The dotted line represents pre-simulated energy consumption without access to storage systems, while the green band illustrates the aggregated charging and discharging behaviour of different agents. The black line showcases the electricity consumption of the DDPG agent when employing its trained policy to control the battery.

In conclusion, the central agent approach demonstrates effectiveness across all problem settings. However, the 50-building scenario reveals areas for improvement, as shown in Figure 11c, where sub optimal policies for some buildings indicate that the policy might be overgeneralising across buildings with more distinct load profiles. However, more research is needed to confirm this claim. Overall, the DDPG agent consistently outperforms other agents and the RBC baseline by a significant margin in all settings.

7.2 Independent multi-agent

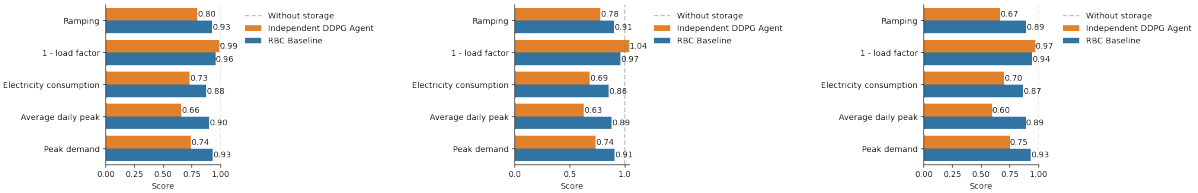
For the independent agent scenario, we trained identical agents using the same hyperparameters and network topologies. The training curves for the independent agents are averaged, and the standard deviation is indicated by the shaded region. Intuitively, the size of the district should affect grid performance of the independent agents. Our results, illustrated in Figure 13, show that this intuition is not reflected by the results, as the independent agents perform similarly across all environment settings. Notably, TD3, SAC, and DDPG perform similarly, with SAC demonstrating the best training performance, achieved after just 6 episodes of training in the 10-building scenario.



(a) District with 5 buildings **(b)** District with 10 buildings **(c)** District with 50 buildings

Figure 13: Performance curves for the benchmarked independent agent approaches. All algorithms were trained until learning performance stops improving. This seems to be at around 6-7 episodes for the DDPG, SAC and TD3 algorithms, and around 40 episodes for the PPO algorithm.

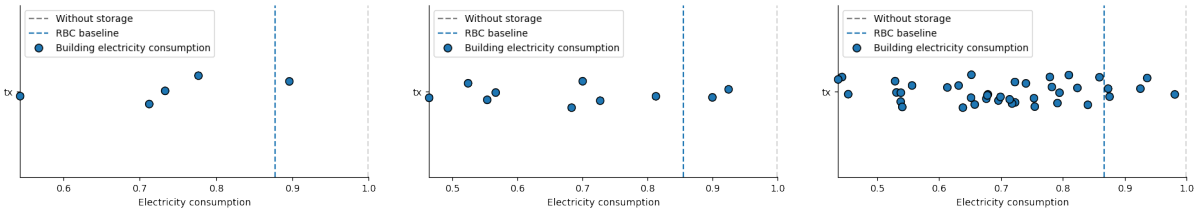
Figure 14 displays the test performance summary for independent agents across all benchmarking environments. The decentralised training approach results in similar performance across different problem settings.



(a) District with 5 buildings **(b)** District with 10 buildings **(c)** District with 50 buildings

Figure 14: Neighbourhood-level KPI summary for the 5, 10 and 50 building CityLearn problems using independent DDPG agents in the testing environment.

Figure 15 illustrates the distribution of electricity consumption KPI scores among individual DDPG agents. Notably, it is evident that not all agents surpass the performance of the RBC baseline. Additionally, the KPI distributions of the single-agent and the independent agents appear relatively similar. This similarity arises because the buildings have different demand profiles, making it inherently harder to optimise net electricity consumption for some buildings compared to others.



(a) District with 5 buildings **(b)** District with 10 buildings **(c)** District with 50 buildings

Figure 15: KPI distribution plot for the electricity consumption of every building in each scenario for the independent DDPG agents. It is clear that in contrast to the central agent, the independent agents manage to find a decent policy for each building.

In Section 7.3, we compare the outcomes of training central agents with those of training multiple individual agents. In Section 7.4, we will discuss the results of the benchmarked CTDE method.

7.3 Central Agent vs. Independent Agents

In this section, we compare the performance of central agent algorithms with independent agent algorithms across various environment settings. In Section 7.1, we demonstrated that DDPG performs best among central agent scenarios in all three environments, although its performance dips slightly in the 50-building scenario (Figure 7). Conversely, Section 7.2 examines results for independent agent approaches, where independent DDPG consistently outperforms other agents across all environments. Of particular interest is whether the central agent method maintains superiority over individual agent methods in larger districts, which will be explored further in this section.

Figure 16 depicts the training curves of both central agent methods and independent agents in a unified plot. To facilitate comparison despite differing training durations as discussed in Section 7.1 and Section 7.2, we aligned the x-axis as training progress (%) by aggregating the training data of the central agents. The figure demonstrates that in this scenario, both central and independent DDPG agents exhibit the best performance, followed by the independent SAC agent and both central and independent TD3 agents.

In Table 5, the KPI scores at test time after training are presented. The test scores correlate with the training curves, indicating that the models do not overfit on the training data. In addition, the table shows that RL methods significantly outperform the RBC baseline.

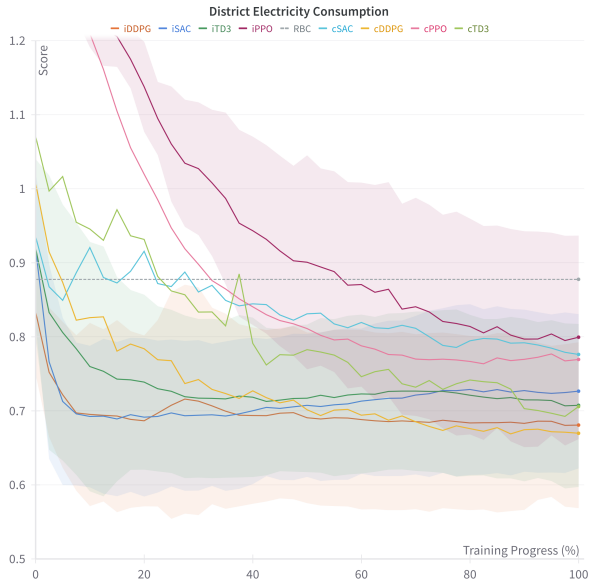


Figure 16: Comparison of central agent and independent agent approaches in the 5-building environment. Among the benchmarked algorithms, central agent DDPG (cDDPG) and independent agent DDPG (iDDPG) demonstrate superior performance, with TD3 approaches following closely. The choice of RL method appears to have more influence on the performance than the decision between central or independent agent approaches.

Figure 17 shows the training performance of all benchmarked methods on the 50 building scenario. In this setting, the independent agents, specifically iDDPG and iTD3 show the most promise. However, during test time, the cDDPG agent performs similarly to the iDDPG agent across all evaluation KPI scores (0.76 versus 0.74). Additionally, as depicted in Figure 14c, both cDDPG and iDDPG agents both score 0.70 for the electricity consumption KPI. This seems to indicate that the independent agents slightly overfit on the training data.

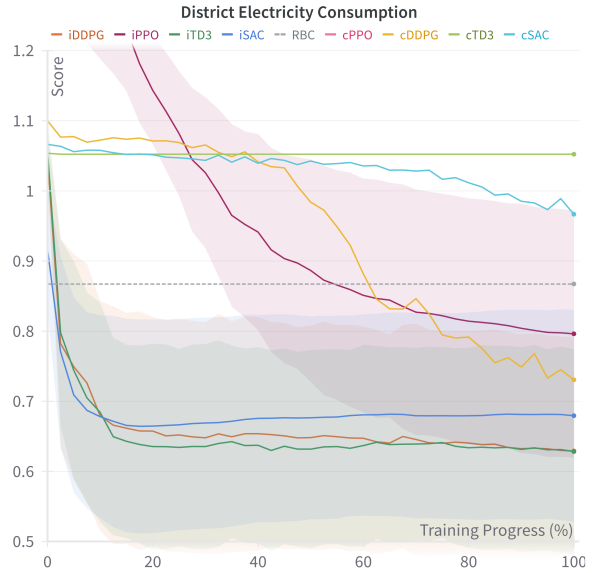


Figure 17: Comparison of central agent and independent agent approaches in the 50-building environment. In this setting, the individual agent approaches seem to outperform the central agent approaches. Independent TD3 and DDPG score the best, followed by independent SAC and central agent DDPG.

Table 5 shows the complete averaged KPI scores at test time for all benchmarked algorithms. In the smaller district sizes, the central DDPG agent outperforms all other agents and the RBC baseline. In the larger district, the independent DDPG agent scores the best across all KPIs. However, the performance is very similar of that of the central DDPG agent.

Table 5: Averaged KPI scores for each algorithm on every district size. For smaller district sizes, central agent DDPG outperforms all other algorithms. For a larger district, the independent DDPG agents slightly outperform the central DDPG agent. The other independent methods do not score as high, this is mainly due to high ramping costs.

Algorithms	**District size**		
	5	10	50
iDDPG	0.78	0.78	**0.74**
iTD3	0.79	0.77	0.91
iSAC	0.81	0.75	0.92
iPPO	0.84	0.82	0.95
cDDPG	**0.73**	**0.71**	0.76
cTD3	0.78	0.80	1.02
cSAC	**0.73**	0.84	0.79
cPPO	0.82	0.84	1.01
RBC	0.92	0.91	0.90

While the results suggest that the independent agent approach might be preferable for larger district sizes, Figure 18 illustrates that the central agent achieves better scores on the electricity consumption KPI during testing. Although the independent agent approach manages to reduce electricity consumption by a larger amount for a small number of buildings, on average, it appears that the central agent optimises building policies more effectively. There are a few outliers where performance is comparable to not using energy storage at all, and one case where performance is significantly worse than that. The training curve indicates that further improvements can still be made with the central agent approach, whereas the performance curves for individual agents plateau early, as depicted in Figure 13.

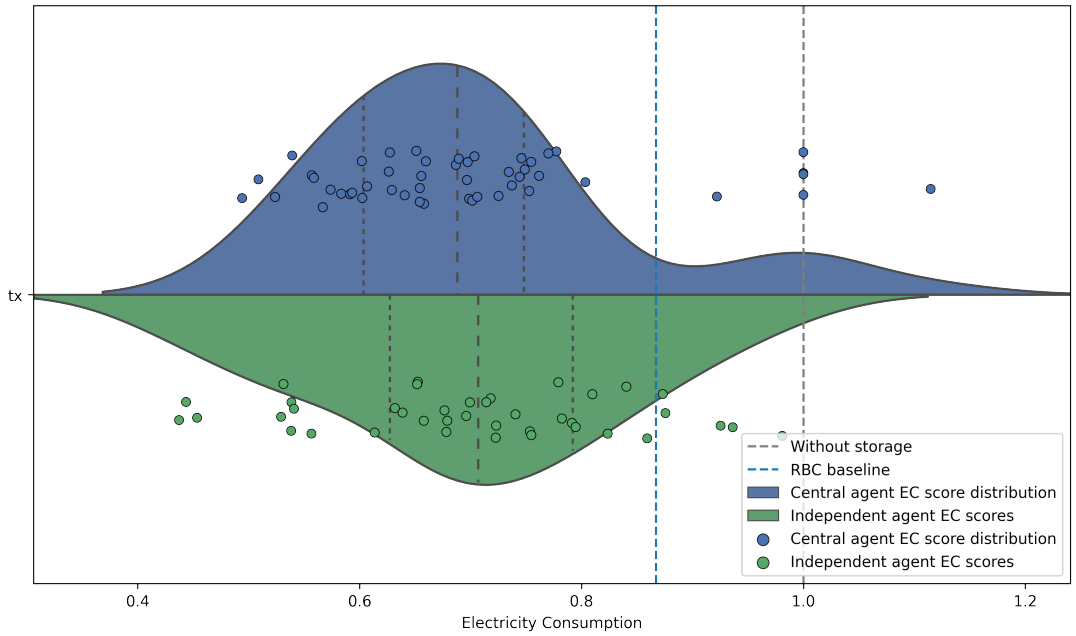


Figure 18: Violin plot illustrating the comparison of the distribution of the electricity consumption KPI scores for the central agent and the independent agent. The central agent seems to achieve more consistent and optimised results across the district, except from a few outliers.

7.4 CTDE Agent

In this research, we implemented the MAPPO algorithm introduced in Section 7.4 as the benchmarked CTDE controller. Training the MAPPO algorithm on our dataset over 1M time steps with five buildings reveals that the model does not perform better than the baseline of doing nothing. This may be due to the large observation space for the value function, which considers a concatenation of all the building observations. Figure 19 shows the performance of the MAPPO agent during 1M time steps. Further research is needed to find the correct hyperparameters that enable the MAPPO agent to learn this task.

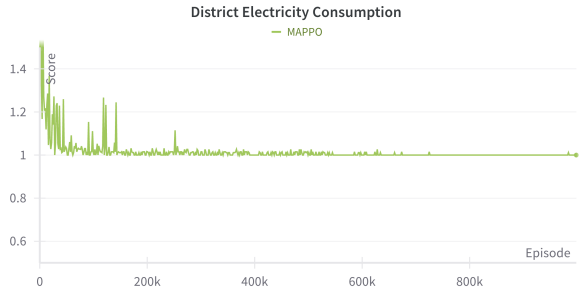


Figure 19: Performance curve of the MAPPO algorithm on the easiest 5 building task. The agent is not able to learn a policy that is better than the baseline of doing no DR.

8 Discussion

This research has brought several important results to light. The first surprising finding is that the DDPG agents outperform the other agents across all settings. We discussed all methods in Section 6, first introducing DDPG, which was proposed in 2015. Next we introduced TD3 and SAC, both released concurrently in 2018, which were proposed as improvements on the DDPG algorithm. Both methods were benchmarked against DDPG, outperforming it on several popular MuJoCo environment benchmarks (Fujimoto et al., 2018) (Haarnoja et al., 2018). These excellent results might explain why, in earlier studies, the SAC algorithm is predominantly chosen to solve problems in the CityLearn environment, as indicated in Section 2. This research suggests that the DDPG algorithm could be better suited for addressing these kinds of problems. Understanding this result requires exploring why the DDPG algorithm might be particularly well-suited for addressing these specific challenges.

The benchmark environments used for SAC and TD3 in their original papers differ significantly from CityLearn. The MuJoCo environment features settings settings with sparse rewards. In contrast, the dynamics of the CityLearn simulation exhibits low stochasticity and provides a dense reward signal at each time step, potentially aiding DDPG in learning the problem more effectively.

TD3 employs the clipped double Q-learning technique along with delayed policy updates (Section 6.4.2) aimed to mitigate overestimation bias, leading to conservative network updates. However, in environments like CityLearn with dense rewards and determinism, overestimation bias might be less of an issue. These conservative network updates could be actually detrimental for learning in the CityLearn environment. If overestimation is not an issue, DDPG should learn a good policy more quickly.

SAC also uses target networks and the clipped double Q trick, but also utilises a stochastic policy using the maximum entropy framework, which encourages more exploration by the agent (Section 6.4.3). This exploration strategy may lead the agent to exploit high-reward strategies less frequently. A more greedy strategy could actually be advantageous for learning a policy in this particular environment.

The difference of the methods is especially evident in the 50-building scenario, where only the DDPG agent manages to learn a good policy whereas the performance curves during training for other agents appear to plateau. Further research is needed to confirm this observation, as SAC and TD3 algorithms may

require more time to converge to a good policy due to their conservative updates and higher exploration. Given sufficient time, SAC and TD3 might eventually be able to solve the environment. Achieving this might also require hyperparameter tuning.

Finally, an important result of this research is the discovery that the central agent approach is more effective than earlier research (discussed in Section 2) suggests. The independent DDPG agent slightly outperformed the central DDPG agent only in the large district size scenario. However, as mentioned in Section 7.3, the testing results seem to indicate that the independent agents slightly overfit on the training data, whereas the training curve of the central agent (Figure 14c) indicates that there might still be additional performance gains achievable with an even longer training schedule. Additionally, Figure 18 highlights there is room left for improvement in the central agent approach.

8.1 Limitations

In the results of our experiments, it is evident that while the reward function optimises net energy usage, the other KPIs, except for the 1 - load factor, are also optimised concurrently. This reward function, introduced by Nweye et al., is one of the default reward functions in the CityLearn environment (Vázquez-Canteli et al., 2019) and was used in their showcase of this environment in (Nweye et al., 2023b). In this work, this behaviour was already indicated, prompting us to choose for this reward function. However, further customisation of this reward function is encouraged in the CityLearn challenges Section 4.4 and could increase the performance of RL agents across all KPIs even more. An interesting approach could be to design a reward function that optimises all KPIs together.

All benchmarked algorithms were trained using simulation data from June and July, with testing conducted using simulated data from August, all within the summer season as detailed in Section 5.2. As shown in Figure 5, temperatures during these months are relatively consistent, suggesting that optimal DHW control strategies might also be similar. Further research could involve training the agent on weeks spanning the entire year with unevenly numbered weeks for training and evenly numbered weeks for testing. Using the trained agents in this study to control a building throughout the entire year might result in poorer performance, especially during winter months when conditions differ significantly. Increasing the amount of training data could improve the performance of RL agents. However, controlling a building throughout the entire year presents a greater challenge due to varying conditions and load profiles. Therefore, developing a more complex strategy capable of adapting to these changes may require more training time than learning a policy that performs well only in the summer.

In Appendix A.2, we discussed the neural network architectures of each benchmarked algorithm. For this research, we retained the default network architectures from the Stable Baselines3 code base, which are based on the original proposals of these algorithms. However, using differently sized network architectures could lead to unfair comparisons among the algorithms. While PPO uses a 2x64 architecture, SAC, DDPG, and TD3 employ relatively larger architectures. This might have caused the PPO agent to underfit the data. Scaling up the network architecture proved successful for the DDPG agent in solving the larger 50-building environment, but did not change the performance of the other agents significantly. In Section 6.5.2, we noted that overfitting due to a large network size is rarely an issue. Further research is needed to verify whether the smaller network size of the agents causes them to underfit the data due to limited capacity.

8.2 Future work

As indicated by this study, DDPG outperformed the PPO agent. Therefore, a promising direction for future research would be to evaluate the performance of a MADDPG agent (Lowe et al., 2017). MADDPG extends the DDPG algorithm within the CTDE framework, employing a shared centralised critic network alongside individual policy networks for the agents.

Further research is required to train central agent approaches on large-scale districts. As suggested in Section 8, as the distribution comparison of the individual building net electricity consumption KPIs depicted in Figure 18 shows, more gains could be achieved using the central agent approach. This work

would entail research on the trained policy of the outliers of the distribution, and trying to mitigate this issue.

Although we experimented with a larger neural network architecture, conducting a thorough hyperparameter search is needed for effectively comparing the algorithms on the larger scale setting. Adjusting various hyperparameters could potentially enable the TD3 and SAC agents to successfully address the problem.

More research can be done comparing central agents against individual agents using an in-depth hyperparameter optimisation search for both methods. Since the central agent method showed significant benefits from the larger network architecture, this modification should be explored for both individual and central agents. However, implementing this for individual agents would require considerably more computational resources that scale with the district size.

The selected dataset includes data for simulating up to 100 buildings. However, due to computational complexity, districts with 100 buildings were not included in this study. It would be interesting to investigate whether the central agent approach remains effective in such scenarios and whether scaling up the neural network size even more is necessary.

The benchmarked CTDE method (Section 7.4) was not showcased well in this research, not converging to a good policy. Earlier research has shown that the MAPPO agent, in combination with forecasting models was able to achieve the highest score on the 2022 CityLearn challenge Section 4.4. Further research could investigate performance improvements for the MAPPO algorithm through hyperparameter optimisation.

Reflecting on the findings of this study, the most promising research direction would be to reproduce the results with a simulation that includes more extensive training data spanning multiple years. Additionally, conducting a thorough hyperparameter search for all benchmarked algorithms and optimising neural network architectures for all methods are necessary to ensure a fair comparison. Furthermore, expanding the research to simulate environments with 100 buildings would provide a more comprehensive evaluation of the scalability of different methods. While RL seems to be a promising approach for solving DR problems in simulation, it still needs to be tested thoroughly in real-world environments.

9 Conclusion

To conclude, the primary finding of this research suggests that DDPG may offer significant advantages over the commonly used SAC agent in related works (as discussed in Section 2.3) for solving demand response problems in the CityLearn environment. This is a surprising result, given that SAC and TD3 were originally proposed as improvements over DDPG and have shown better performance in other benchmark environments. The success of DDPG in this context may be attributed to the dense reward signals and low stochasticity of the CityLearn environment, which favour the learning characteristics of DDPG. Compared to the traditionally used RBC for demand response problems, RL approaches demonstrate significant performance improvements over this baseline, thereby confirming RL as a promising approach for addressing this challenge.

Furthermore, the findings of this study indicate that central agent approaches are potentially be better suited for solving the demand response problem in CityLearn, likely because they can leverage the complete observation space of the environment. This advantage holds especially true in smaller district sizes, while in larger districts they perform comparably to independent agents when evaluated on the KPIs. During training, the performance of the independent agents appears superior, but this discrepancy may indicate an overgeneralising of the independent agents on the training data.

Our results suggest that central agent approaches show potential for further improvement. However, in practical applications, independent approaches might be preferred due to their potential for decentralised execution, eliminating the need to connect all buildings to a central processing unit. Choosing between central and independent agents should depend on the size of real-world districts: central agents may be

more beneficial for smaller districts, while independent agents may be preferable for larger districts due to easier deployment and comparable performance.

Looking forward, CTDE methods offer a promising approach by combining the benefits of centralised training with decentralised execution using individual policy networks. A promising research venue would be to compare the performance of a MADDPG agent implemented within the CTDE framework against its central and independent agent counterparts.

In summary, this study underscores the potential of RL approaches in demand response strategies within urban energy management. Specifically, we point out the DDPG algorithm as the most promising choice for further research and highlight the strengths and weaknesses of central and independent agent approaches. Future studies should focus on extending the application of the DDPG algorithm to similar continuous energy management challenges and the implementation of RL agents in real-world environments.

References

Akorede, M. F., Hizam, H., and Pouresmaeil, E. (2010). Distributed energy resources and benefits to the environment. *Renewable and Sustainable Energy Reviews*, 14(2):724–734.

Almilaify, Y., Nweye, K., and Nagy, Z. (2023). SCALEX: SCALability EXploration of Multi-Agent Reinforcement Learning Agents in Grid-Interactive Efficient Buildings. In *Proceedings of the 10th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, pages 261–264.

Bellman, R. (1966). Dynamic programming. *Science*, 153(3731):34–37.

Biagioni, D., Zhang, X., Wald, D., Vaidhynathan, D., Chintala, R., King, J., and Zamzam, A. S. (2022). PowerGridWorld: A framework for multi-agent reinforcement learning in power systems. In *Proceedings of the Thirteenth ACM International Conference on Future Energy Systems*, pages 565–570.

Cao, D., Hu, W., Zhao, J., Zhang, G., Zhang, B., Liu, Z., Chen, Z., and Blaabjerg, F. (2020). Reinforcement learning and its applications in modern power and energy systems: A review. *Journal of Modern Power Systems and Clean Energy*, 8(6):1029–1042.

Chen, B., Yao, W., Francis, J., and Bergés, M. (2020). Learning a distributed control scheme for demand flexibility in thermostatically controlled loads. In *2020 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, pages 1–7.

Crawley, D., Pedersen, C., Lawrie, L., and Winkelmann, F. (2000). EnergyPlus: Energy simulation program. *Ashrae Journal*, 42:49–56.

Dhamankar, G., Vazquez-Canteli, J. R., and Nagy, Z. (2020). Benchmarking multi-agent deep reinforcement learning algorithms on a building energy demand coordination task. In *Proceedings of the 1st International Workshop on Reinforcement Learning for Energy Management in Buildings & Cities*, pages 15–19.

Forbu, A. (2022). Citylearn challenge - going below 1.0 score with stablebaselines3. https://www.aicrowd.com/showcase/going-below-1-0-score-with-stablebaseline3.

François-Lavet, V., Taralla, D., Ernst, D., and Fonteneau, R. (2016). Deep reinforcement learning solutions for energy microgrids management. In *European Workshop on Reinforcement Learning (EWRL 2016)*.

Fujimoto, S., Hoof, H., and Meger, D. (2018). Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pages 1587–1596.

Gils, H. C. (2014). Assessment of the theoretical demand response potential in europe. *Energy*, 67:1–18.

Haarnoja, T., Moran, B., Lever, G., Huang, S. H., Tirumala, D., Humplik, J., Wulfmeier, M., Tunyasuvunakool, S., Siegel, N. Y., Hafner, R., et al. (2024). Learning agile soccer skills for a bipedal robot with deep reinforcement learning. *Science Robotics*, 9(89):eadi8022.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pages 1861–1870.

Intelligent Environments Lab & AIcrowd (2022). The CityLearn Challenge 2022. https://www.aicrowd.com/challenges/neurips-2022-citylearn-challenge.

Intelligent Environments Lab & AIcrowd (2023). NeurIPS 2023 CityLearn Challenge. https://www.aicrowd.com/challenges/neurips-2023-citylearn-challenge/problems/control-track-citylearn-challenge.

Kalashnikov, D., Irpan, A., Pastor, P., Ibarz, J., Herzog, A., Jang, E., Quillen, D., Holly, E., Kalakrishnan, M., Vanhoucke, V., et al. (2018). Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on robot learning*, pages 651–673. PMLR.

Kathirgamanathan, A., Twardowski, K., Mangina, E., and Finn, D. P. (2020). A Centralised Soft Actor Critic Deep Reinforcement Learning Approach to District Demand Side Management through CityLearn. In *Proceedings of the 1st International Workshop on Reinforcement Learning for Energy Management in Buildings & Cities*, pages 11 – 14.

Khattar, V. and Jin, M. (2022). Winning the CityLearn challenge: adaptive optimization with evolutionary search under trajectory-based guidance. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 14286–14294.

Konda, V. and Tsitsiklis, J. (1999). Actor-critic algorithms. *Advances in neural information processing systems*, volume 12.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2015). Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Littman, M. L. (1994). Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994*, pages 157–163. Elsevier.

Lowe, R., WU, Y., Tamar, A., Harb, J., Abbeel, P., and Mordatch, I. (2017). Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Luo, J., Paduraru, C., Voicu, O., Chervonyi, Y., Munns, S., Li, J., Qian, C., Dutta, P., Davis, J. Q., Wu, N., et al. (2022). Controlling commercial cooling systems using reinforcement learning. *arXiv preprint arXiv:2211.07357*.

Maji, D., Shenoy, P., and Sitaraman, R. K. (2022). Carboncast: multi-day forecasting of grid carbon intensity. In *Proceedings of the 9th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, pages 198–207.

Mattsson, S. E., Elmqvist, H., and Otter, M. (1998). Physical system modeling with modelica. *Control engineering practice*, 6(4):501–510.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Nagy, Z., Henze, G., Dey, S., Arroyo, J., Helsen, L., Zhang, X., Chen, B., Amasyali, K., Kurte, K., Zamzam, A., et al. (2023). Ten questions concerning reinforcement learning for building energy management. *Building and Environment*, 241:110435.

Nagy, Z., Vázquez-Canteli, J. R., Dey, S., and Henze, G. (2021). The Citylearn Challenge 2021. In *Proceedings of the 8th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, page 218–219. Association for Computing Machinery.

Narayanamurthy, R., Handa, R., Tumilowicz, N., Herro, C., and Shah, S. (2016). Grid integration of zero net energy communities. *ACEEE Summer Study Energy Effic. Build.*

Neukomm, M., Nubbe, V., and Fares, R. (2019). Grid-interactive efficient buildings technical report series: Overview of research challenges and gaps.

Nweye, K., Kaspar, K., Buscemi, G., Pinto, G., Li, H., Hong, T., Ouf, M., Capozzoli, A., and Nagy, Z. (2023a). CityLearn v2: An OpenAI Gym environment for demand response control benchmarking in grid-interactive communities. In *Proceedings of the 10th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, page 274–275. Association for Computing Machinery.

Nweye, K., Kaspar, K., Buscemi, G., Pinto, G., Li, H., Hong, T., Ouf, M., Capozzoli, A., and Nagy, Z. (2023b). A framework for the design of representative neighborhoods for energy flexibility assessment in citylearn. In *Building Simulation 2023*, volume 18, pages 3346–3353. IBPSA.

Nweye, K., Liu, B., Stone, P., and Nagy, Z. (2022a). Real-world challenges for multi-agent reinforcement learning in grid-interactive buildings. *Energy and AI*, 10:100202.

Nweye, K., Nagy, Z., Mohanty, S., Chakraborty, D., Sankaranarayanan, S., Hong, T., Dey, S., Henze, G., Drgona, J., Lin, F., et al. (2022b). The CityLearn Challenge 2022: Overview, results, and lessons learned. *NeurIPS 2022 Competition Track*, pages 85–103.

Nweye, K., Sankaranarayanan, S., and Nagy, Z. (2023c). MERLIN: Multi-agent offline and transfer learning for occupant-centric operation of grid-interactive communities. *Applied Energy*, 346:121323.

Pinto, G., Kathirgamanathan, A., Mangina, E., Finn, D. P., and Capozzoli, A. (2022). Enhancing energy management in grid-interactive buildings: A comparison among cooperative and coordinated architectures. *Applied Energy*, 310:118497.

Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., and Dormann, N. (2021). Stable-Baselines3: Reliable Reinforcement Learning Implementations. *Journal of Machine Learning Research*, 22(268):1–8.

Rangu, S. K., Lolla, P. R., Dhenuvakonda, K. R., and Singh, A. R. (2020). Recent trends in power management strategies for optimal operation of distributed energy resources in microgrids: A comprehensive review. *International Journal of Energy Research*, 44(13):9889–9911.

Rashid, T., Samvelyan, M., De Witt, C. S., Farquhar, G., Foerster, J., and Whiteson, S. (2020). Monotonic value function factorisation for deep multi-agent reinforcement learning. *Journal of Machine Learning Research*, 21(178):1–51.

Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., Lillicrap, T., and Silver, D. (2020). Mastering Atari, Go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. (2015). Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016). Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. (2014). Deterministic policy gradient algorithms. In *International conference on machine learning*, pages 387–395. Pmlr.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. (2017). Mastering the game of go without human knowledge. *nature*, 550(7676):354–359.

Sinsel, S. R., Riemke, R. L., and Hoffmann, V. H. (2020). Challenges and solution technologies for the integration of variable renewable energy sources—a review. *renewable energy*, 145:2271–2285.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Sutton, R. S., McAllester, D., Singh, S., and Mansour, Y. (1999). Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12.

Tan, X., Li, Q., and Wang, H. (2013). Advances and trends of energy storage technology in microgrid. *International Journal of Electrical Power Energy Systems*, 44:179–191.

Towers, M., Terry, J. K., Kwiatkowski, A., Balis, J. U., de Cola, G., Deleu, T., Goulão, M., Kallinteris, A., KG, A., Krimmel, M., Perez-Vicente, R., Pierré, A., Schulhoff, S., Tai, J. J., Tan, A. J. S., and Younis, O. G. (2022). Gymnasium. Available at https://gymnasium.farama.org/.

Van Hasselt, H., Guez, A., and Silver, D. (2016). Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30.

Vazquez-Canteli, J. R., Dey, S., Henze, G., and Nagy, Z. (2012). Citylearn: Standardizing research in multi-agent reinforcement learning for demand response and urban energy management. *arXiv preprint arXiv:2012.10504*.

Vázquez-Canteli, J. R., Dey, S., Henze, G., and Nagy, Z. (2020). The Citylearn Challenge 2020. In *Proceedings of the 7th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, pages 320–321.

Vazquez-Canteli, J. R., Henze, G., and Nagy, Z. (2020). MARLISA: Multi-agent reinforcement learning with iterative sequential action selection for load shaping of grid-interactive connected buildings. In *Proceedings of the 7th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, page 170–179. Association for Computing Machinery.

Vázquez-Canteli, J. R., Kämpf, J., Henze, G., and Nagy, Z. (2019). Citylearn v1.0: An openai gym environment for demand response with deep reinforcement learning. In *Proceedings of the 6th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, page 356–357. Association for Computing Machinery.

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., et al. (2019). Grandmaster level in starcraft ii using multi-agent reinforcement learning. *nature*, 575(7782):350–354.

Watkins, C. J. and Dayan, P. (1992). Q-learning. *Machine learning*, 8:279–292.

Wilcox, S. and Marion, W. (2008). Users manual for tmy3 data sets. Technical Report NREL/TP-581-4315, National Renewable Energy Laboratory, Golden, CO, United States.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256.

Wilson, E. J. (2017). Resstock-targeting energy and cost savings for us homes. Technical report, National Renewable Energy Laboratory, Golden, CO, United States.

Wilson, E. J., Parker, A., Fontanini, A., Present, E., Reyna, J. L., Adhikari, R., Bianchi, C., CaraDonna, C., Dahlhausen, M., Kim, J., et al. (2022). End-use load profiles for the us building stock: Methodology and results of model calibration, validation, and uncertainty quantification. Technical report, National Renewable Energy Laboratory, Golden, CO, United States.

Yu, C., Velu, A., Vinitsky, E., Gao, J., Wang, Y., Bayen, A., and Wu, Y. (2022). The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in Neural Information Processing Systems*, 35:24611–24624.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. (2021). Understanding deep learning (still) requires rethinking generalization. *Commun. ACM*, 64(3):107–115.

A Appendix

A.1 Complete CityLearn Observation Space

Table 6 shows the complete observation space for CityLearn v1.8.0, which was used in this research. Central agents methods use the observations that are indicated as 'active' and observations that are the same for every building in the district are called 'shared' and only appear once in the observation of a central agent.

Table 6: Overview of observation parameters in CityLearn v1.8.0, detailing active variables utilised by RL agents and shared observations across buildings.

Observation	Active	Shared in Central Agent
Month	No	Yes
Hour	Yes	Yes
Day Type	Yes	Yes
Daylight Savings Status	No	Yes
Indoor Temperature (C)	No	No
Average Unmet Cooling Setpoint Difference (C)	No	No
Indoor Relative Humidity (%)	No	No
Equipment Electric Power (kWh)	Yes	No
DHW Heating (kWh)	Yes	No
Cooling Load (kWh)	No	No
Heating Load (kWh)	No	No
Solar Generation (W/kW)	Yes	No
Outdoor Drybulb Temperature (C)	Yes	Yes
Outdoor Relative Humidity (%)	No	Yes
Diffuse Solar Radiation (W/m^2)	No	Yes
Direct Solar Radiation (W/m^2)	Yes	Yes
6h Outdoor Drybulb Temperature (C)	Yes	Yes
12h Outdoor Drybulb Temperature (C)	Yes	Yes
24h Outdoor Drybulb Temperature (C)	No	Yes
6h Outdoor Relative Humidity (%)	No	Yes
12h Outdoor Relative Humidity (%)	No	Yes
24h Outdoor Relative Humidity (%)	No	Yes
6h Diffuse Solar Radiation (W/m^2)	No	Yes
12h Diffuse Solar Radiation (W/m^2)	No	Yes
24h Diffuse Solar Radiation (W/m^2)	No	Yes
6h Direct Solar Radiation (W/m^2)	Yes	Yes
12h Direct Solar Radiation (W/m^2)	Yes	Yes
24h Direct Solar Radiation (W/m^2)	No	Yes
Carbon Intensity	No	Yes
Non-shiftable Load	Yes	No
Cooling Storage SoC	No	No
Heating Storage SoC	No	No
DHW Storage SoC	Yes	No
Electrical Storage SoC	Yes	No
Net Electricity Consumption	Yes	No
Electricity Pricing	No	No
Electricity Pricing Predicted 6h	No	No
Electricity Pricing Predicted 12h	No	No
Electricity Pricing Predicted 24h	No	No

A.2 Hyperparameters

Table 7, Table 8, Table 9 and Table 10 show the hyperparameter configurations of the benchmarked algorithms together with their default network architectures.

Table 7: Hyperparameters for Proximal Policy Optimization (PPO)

Hyperparameter	Default Value
learning rate	3×10^{-4}
n steps	2048
batch size	64
gamma	0.99
gae lambda	0.95
clip range	0.2
clip range vf	None
normalize advantage	True
ent coef	0.0
vf coef	0.5
max grad norm	0.5
use sde	False
target kl	None
policy net arch	2 x 64
value net arch	2 x 64

Table 8: Hyperparameters for Soft Actor-Critic (SAC)

Hyperparameter	Default Value
learning rate	3×10^{-4}
buffer size	1,000,000
learning starts	100
batch size	256
tau	0.005
gamma	0.99
policy net arch	2 x 256
value net arch	2 x 256

Table 9: Hyperparameters for Twin Delayed DDPG (TD3)

Hyperparameter	Default Value
learning rate	1×10^{-3}
buffer size	1,000,000
learning starts	100
batch size	256
tau	0.005
gamma	0.99
policy delay	2
target policy noise	0.2
target noise clip	0.5
policy net arch	2 x 400
target policy net arch	2 x 300
value net arch	2 x 400
target value net arch	2 x 300

Table 10: Hyperparameters for Deep Deterministic Policy Gradient (DDPG)

Hyperparameter	**Default Value**
learning rate	1×10^{-3}
buffer size	1,000,000
learning starts	100
batch size	256
tau	0.005
gamma	0.99
policy delay	1
target noise clip	0.0
target policy noise	0.1
policy net arch	2 x 256
value net arch	2 x 256