Eberhard Karls Universität Tübingen
Mathematisch-Naturwissenschaftliche Fakultät
Wilhelm-Schickard-Institut für Informatik

Master Thesis Computer Science

Uncertainty-Aware Reinforcement Learning for Demand Response in Energy Systems

Ludwig Bald

January 31, 2023

Reviewers

Dr. Nicole Ludwig
Wilhelm-Schickard-Institut für Informatik
Universität Tübingen

Jun. Prof. Dr.-Ing.
Setareh Maghsudi
Wilhelm-Schickard-Institut für Informatik
Universität Tübingen

Bald, Ludwig:
Uncertainty-Aware Reinforcement Learning for Demand Response in Energy Systems
Master Thesis Computer Science
Eberhard Karls Universität Tübingen
Thesis period: July 2022-January 2023

Abstract

The electrical grid is increasingly dominated by renewable energy. Therefore, demand needs to flexibly respond to changes in the amount of available electricity. Buildings play a key role in this regard: They consume a large amount of energy for space heating and cooling as well as providing hot water. Intelligently controlled, they can flexibly consume energy when supply is available.

Distributional Reinforcement Learning is a class of machine learning algorithms that can efficiently explore an RL environment and learn a control policy by explicitly modeling uncertainty. Among those, UA-DQN is an algorithm that explicitly estimates epistemic uncertainty, caused by a lack of data, and aleatoric uncertainty, caused by the environment's inherent stochasticity.

I evaluate the performance of UA-DQN on a custom RL environment for building energy management based on the 2022 CityLearn challenge. After tuning the hyperparameters, I evaluate the algorithm against a rule-based baseline and two variants of DQN, using ϵ-greedy and Boltzmann (Softmax) action selection. All three algorithms make use of the battery, but none outperforms the baseline. UA-DQN requires fewer environment interactions to arrive at a superior policy than both DQN variants, but demands more computational resources.

Zusammenfassung

Da die Stromproduktion zunehmend auf erneuerbare Energien umgestellt wird, muss die Nachfrage flexibel auf Veränderungen der verfügbaren Produktionskapazität reagieren. Gebäude spielen in dieser Hinsicht eine wichtige Rolle: Sie verbrauchen einen großen Teil der Energie für Heizung und Kühlung sowie für Warmwasser. Intelligent gesteuert können sie flexibel dann Energie verbrauchen, wenn sie verfügbar ist.

Distributional Reinforcement Learning ist eine Klasse von Algorithmen des maschinellen Lernens, die durch explizite Modellierung von Unsicherheit eine RL-Umgebung effizient erkunden und eine Steuerungspolitik erlernen können. UA-DQN ist ein Algorithmus, der explizit die epistemische Unsicherheit, die durch einen Mangel an Daten verursacht wird, und die aleatorische Unsicherheit, die durch die inhärente Stochastizität der Umgebung verursacht wird, schätzt.

Ich bewerte die Leistung von UA-DQN in einer benutzerdefinierten RL-Umgebung für das Energiemanagement von Gebäuden auf der Grundlage der CityLearn Challenge 2022. Ich evaluiere den Algorithmus mit optimierten Hyperparametern im Vergleich zu einer regelbasierten Baseline und zwei Varianten von DQN, die ϵ-greedy und Boltzmann (Softmax) zur Aktionsauswahl verwenden. Alle drei Algorithmen nutzen den Stromspeicher, aber keiner übertrifft die Baseline. UA-DQN benötigt weniger Interaktionen mit der Umgebung, um zu einer den DQN-Varianten überlegenen Strategie zu gelangen, erfordert aber mehr Rechenressourcen.

Acknowledgements

First and foremost, I would like to express my gratitude to Nicole Ludwig for being a very approachable, kind and helpful primary supervisor. I would also like to thank Francesco Chini for sharing his knowledge of the dark art of Reinforcement Learning.

I am grateful to the entire MLSES group for providing such a welcoming and social atmosphere. I want to thank my officemates and friends for keeping me sane throughout the sometimes stressful phases of this project. Finally, of course, I want to thank Toni for everything!

Contents

Chapter 1

Introduction

Climate Change is the global challenge of our lifetime. Carbon introduced into the atmosphere when burning fossil fuels for human needs causes global warming, destabilizing the climate, ecosystems, and societies around the world. In the Paris Agreement of 2015, governments have committed to an ambitious goal of drastically reducing carbon emissions to keep global warming from increasing beyond 2 °C, compared to 1990. The latest report by the Intergovernmental Panel on Climate Change urges governments to take stronger actions, or their previous commitment will not be reached (Pörtner et al., 2022). A key strategy for reducing carbon emissions from a range of sources is the combination of two measures: The first step is to electrify current processes that use fossil fuels, like replacing gas-fired furnaces with heat pumps. The second step is to replace carbon-intensive electricity generation with renewable options like solar and wind power.

While much better for the natural environment, renewable sources of energy pose a challenge for a grid built for fossil fuels: Unlike fossil-fuelled power plants, renewable power production depends on the weather, and it can not react flexibly to changes in demand. As the share of installed renewable sources of electricity continues to grow, the reliability of the electricity supply will go down.

In order to keep the grid stable, supply and demand must always be in balance. Before the green transition, this was achieved by flexible power generation: When demand was high, electricity producers were able to react and increase production. This was incentivized by a complex and tightly regulated electricity market. As the share of renewable power increases, fossil-fuelled electricity producers remain the only market participants that can flexibly react to changes in renewable production and demand for electricity. When phasing out fossil-fuelled power generation, this flexibility needs to be provided by different parts of the system.

There are dedicated electricity storage facilities that can react very quickly to stabilize the grid by storing and releasing energy as needed. Grid-scale storage in Europe mainly consists of hydropower, which has been installed where mountainous geography allows, and capacity is limited (Gimeno-Gutiérrez and Lacal-Arántegui, 2015). More expensive battery-powered storage facilities are slowly being built, but are largely not cost-efficient in the current economic setting. Therefore, there is a need for *Demand Response*, or the ability of demand to flexibly respond to available electricity supply.

Buildings use energy mainly for heating and cooling air and water supply. Today, they are responsible for 17.5% of global carbon emissions (Ritchie et al., 2020). On the other hand, buildings often contribute to electricity production through photovoltaic panels. New buildings are often equipped with a battery, which enables them to more efficiently use their solar electricity. Building electricity consumption is already largely automated and is therefore a prime candidate for automated demand response.

The *Reinforcement Learning* (RL) family of control algorithms has shown promising performance in many control tasks, most notably video games (for example Mnih et al. (2015)), robotics (Latifi et al., 2020) or wind power generation(Zhang et al., 2020). RL is also being studied for controlling energy systems in buildings. Vazquez-Canteli et al. (2020) introduce CityLearn, which proved a popular simulation framework for applying RL algorithms to building energy storage control. Evaluated algorithms include Soft Actor-Critic (Pinto et al., 2021) and DQN (Schreiber et al., 2020).

Wang and Hong (2020) find that the demand for data is a key challenge for applying RL for building control. Clements et al. (2020) introduce *Uncertainty-Aware Deep Q-Networks* (UA-DQN), a distributional RL algorithm based on DQN (Mnih et al., 2015) that uses explicit uncertainty estimates for approximate Thompson sampling. On a video game task, they show that this allows the algorithm to explore more efficiently, reducing the need for data. UA-DQN's explicit treatment of uncertainty should lead to overall better performance with less need for data, better robustness for novel data, and can even be leveraged for differently risk-aware charging and discharging strategies, all of which are important for building energy management.

In this thesis, I set out to test the performance of UA-DQN on a building battery control task implemented with CityLearn. I compare its performance against two variants of DQN and a rule-based baseline.

This thesis is structured as follows: In chapter 2, I motivate in more detail the need for Automated Demand Response. I introduce the theory of Reinforcement Learning and lay the foundations for the uncertainty-aware algorithm. In chapter 3, I present the uncertainty-aware algorithm, as well as a detailed description of the experimental setup. I present the results in chapter 4. A discussion and a short outlook conclude the thesis.

Chapter 2

Background and Related Work

2.1 The Electrical Grid and Flexibility

The electrical grid is basic infrastructure that enables the function of our modern society. It connects electricity consumers, from private consumers to heavy industry, with producers. Historically, the entire system has been built for reliable large-scale power generation, overwhelmingly fuelled by coal and natural gas, together accounting for 67% of Germany's electricity consumption in 1985, with 27% provided by nuclear energy (Ritchie et al., 2022).

Climate change and the exit from nuclear power require a radical increase in the share of renewable electricity. As of 2021, renewable energy accounts for 40% of electricity consumption in Germany (Ritchie et al., 2022). An overview of the historical development is shown in figure 2.1

While conventional energy generation can flexibly respond to demand, renewable energy depends on the weather. It is therefore intermittent and harder to predict, see figure 2.2. To be able to meet demand, there is a new need for flexible backup power. Natural gas plants are more environmentally friendly than coal plants and can be flexibly turned on and off in a matter of minutes. As a result, the share of electricity powered by natural gas has risen along with renewables.

Electricity demand additionally increases because of the electrification of many processes which previously used fossil fuels directly. Internal combustion engine cars are being replaced by more efficient battery-powered electric cars and heating units that use natural gas or oil are being replaced by much more efficient electric heat pumps. Many heat pumps can also be run in reverse to cool a building, which climate change makes more and more necessary, but this enables additional electricity demand in the summer.

The ongoing energy transition puts a lot of pressure on the grid. On the one hand, the electricity supply is becoming less reactive, while on the other hand, many polluting processes are being electrified, causing additional de-

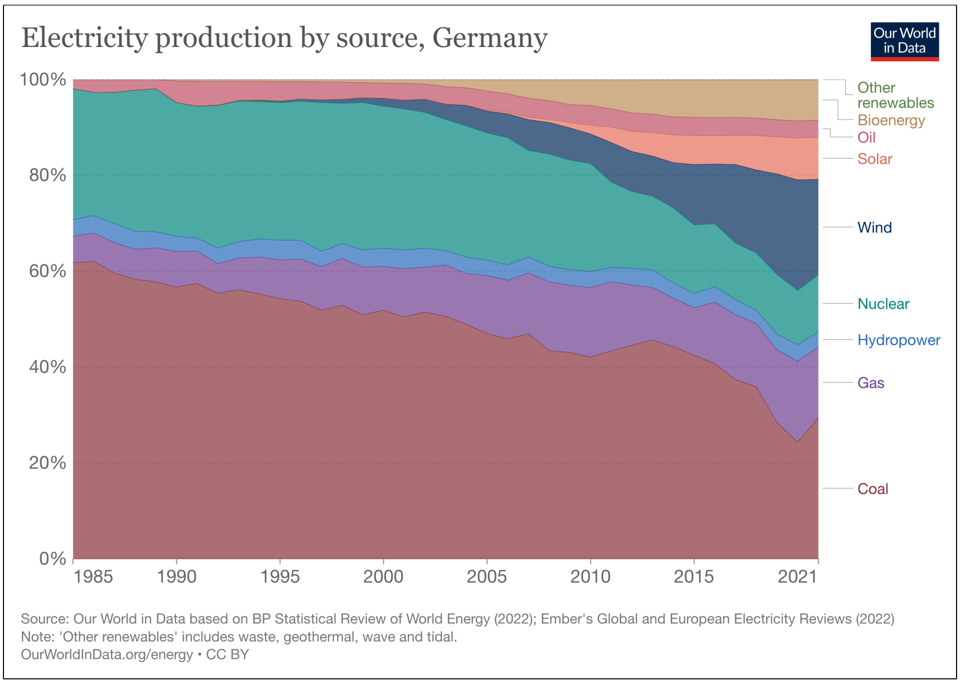

Figure 2.1: The share of renewable electricity has risen significantly from 1985 to 2021, while nuclear and fossil fuels have declined.

mand. During large future demand spikes, the total load on the grid could exceed current capacity. To cope, new transmission lines are being built, a costly and complicated process.

In order to both reduce reliance on fossil fuels and to keep the total load below grid capacity, there is a need for additional flexibility in the system. A traditional source of flexibility has been pumped hydropower storage. When there's unused electricity, it can be stored by pumping water uphill. When needed, water can be released and used to generate electricity. Hydropower storage capacity depends almost entirely on geography and is limited (Gimeno-Gutiérrez and Lacal-Arántegui, 2015).

A different large-scale method of energy storage is batteries, which have only found limited use due to their cost. Hydrogen can also be used as a way to store and even transport energy, with several chemical processes currently being explored. However, hydrogen production capacity is needed for industrial applications first. There is a large need to replace hydrogen generated as a byproduct of the fossil fuel industry.

As flexibility in electricity supply decreases, and large-scale centralized storage facilities remain prohibitively expensive, electricity demand needs to become more flexible.

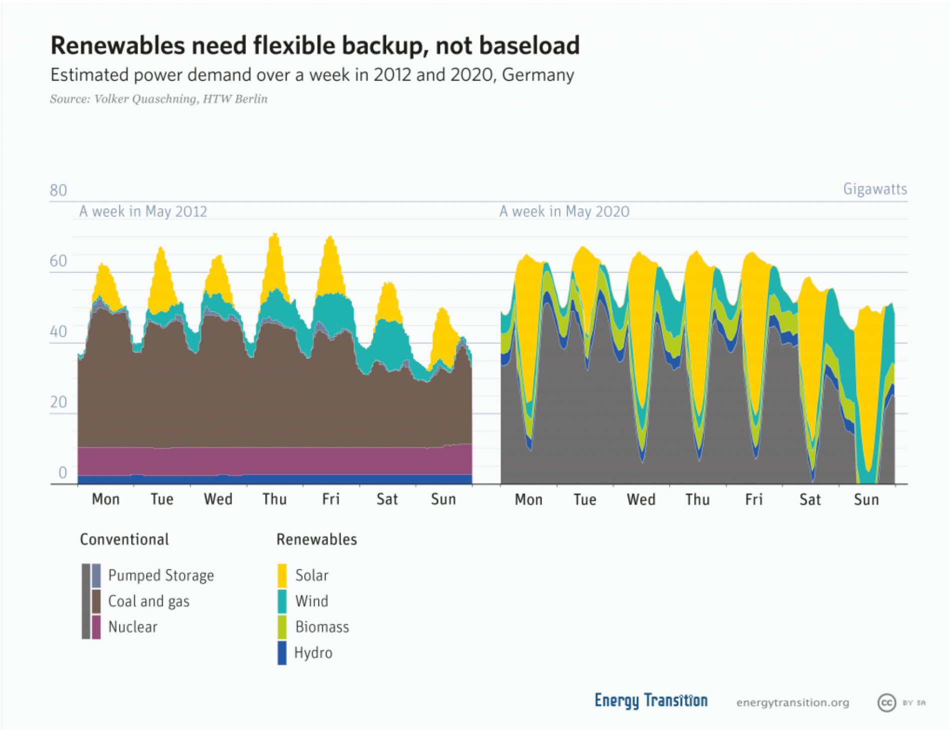

Figure 2.2: Renewable electricity is intermittent and hard to predict precisely. As the share of renewable electricity increases, the electrical grid needs to respond flexibly to meet demand.

2.2 Demand Response

Both the inflexibility of renewable power generation and the limited capacity for centralized flexible energy storage leave one component of the electric system: In a renewable-dominated grid, demand needs to flexibly respond to changes in available supply. In order to stabilize the grid, grid operators already employ schemes to curb demand in case of exceptionally large pressure on the grid. Large industrial consumers are paid in advance for shutting off their processes if needed. As a matter of last resort, rolling blackouts are introduced to curb demand when electricity production can not keep up with consumption. An electric grid designed for renewable energy needs more fine-grained coordination across a larger fraction of demand.

In order to implement consumer demand response, electricity consumers need to be incentivized and able to adapt their processes to available supply. Proposed incentive schemes for demand response include market-based solutions that feature a flexible electricity price and solutions of centralized control that pay out flat rewards for participation, as well as combinations of the two (Khajavi et al., 2011). In order to be able to react to changes in demand, electricity-consuming processes need to be aware of the current and future available supply. This can be a human in the loop, deciding to shift a process to a time with cheaper electricity, or this can be automated. For example, a private prosumer that produces solar electricity might prefer to run their washing machine only on sunny days when electricity is free to them. However, this means the human needs to keep track of the weather and is put under additional cognitive load when planning this.

As stated before, the process of heating is being electrified. While this increases the load on the electric grid, it is also a chance to provide flexibility: When equipped with an intelligent and connected control system, the heating system can flexibly react to changes in price, using hot water tanks or the building itself as heat storage. Similarly, when intelligently automated, the charging process of an electric car or a home battery can react to market conditions.

Technologically, this amounts to a control problem: The controller's goals are to meet certain demands (like ensuring a comfortable living temperature) while minimizing cost. In the context of the larger system, there is the shared goal of coordination between different electricity-consuming processes. This involves both an understanding of the dynamics of the controlled system and an understanding of how prices are likely to change. Different technological approaches can be employed for this, for example Rule-Based Controllers or Model Predictive Control (Li et al., 2021).

2.3 Reinforcement Learning

2.3.1 Reinforcement Learning Fundamentals

Reinforcement Learning (RL) is a feedback-based learning paradigm derived from behavior learning in animals. This section serves as a brief introduction to the topic. Unless where it's stated otherwise, this section is based on Sutton and Barto (2018).

In Reinforcement Learning, there is a clear distinction between the learning agent and the environment. The agent is able to observe the state of the environment and perform an action. In turn, the environment is affected by the action and transitions into a new state according to its stochastic transition dynamics. The environment passes the resulting state and a reward signal back to the agent. Typically, the agent's goal is to select actions that obtain the maximum reward. In an infinite environment, the objective is to maximize the expected value of the total discounted future reward.

The environment specifies the entire reinforcement learning task. Formally, it is a discounted Markov Decision Process (MDP):

$$\text{MDP} = (S, A, R, \gamma, p),$$

where S is the set of possible states, A is the set of possible actions, R is the set of possible rewards, γ is the discount rate and $p(s', r|s, a)$ is the probability distribution that specifies the environment dynamics. It is important to note that an MDP has the Markov Property, i.e. the dynamics depend entirely on the state and action, there is no hidden state. The state space is therefore also the entire observation space.

When modeling a real-world control problem, an MDP necessarily is a simplifying assumption. In reality, state transitions often depend on outside influences or are non-stationary for other reasons. The reward function shapes the agent's behavior. In complex problems, the desired behavior is not obvious. Therefore, designing the reward function is often non-trivial.

2.3.2 Q-Learning

Given an MDP, one algorithm for learning a policy is *Q-Learning*. Q-Learning is a type of Value Learning algorithm. From observed environment interactions, the algorithm builds up an internal representation of action values $q(s, a)$, called *Q-function*. Formally, the Q-function is the expected value of total discounted reward G_t, when starting in state s, taking action a, and then following a policy π.

$$q_\pi(s, a) = \mathbb{E}_\pi \left[G_t | S_t = s, A_t = a\right] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a\right] \quad (2.1)$$

The goal is to approximate the optimal Q-function q_*, which describes the expected return when taking an action and then following the optimal policy π_*.

On the optimal Q-function q_π, the Bellman optimality condition must hold:

$$q_*(s,a) = \mathbb{E}\left[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a')|S_t = s, A_t = a\right] \tag{2.2}$$

Given a Q-function, a greedy policy can be derived:

$$\pi(s) = \max_a(q_*(s,a)) \tag{2.3}$$

Putting the pieces together, it can be shown that the Bellman condition can be turned into an update rule which leads to iterative improvement of q, given an environment interaction:

$$q'(s,a) := (1-\alpha)q(s,a) + \alpha\left(R_{t+1} + \gamma \max_{a'} q_*(s', a')\right), \tag{2.4}$$

where α is the learning rate.

In order to generate all the needed environment interactions, one can use the ϵ-greedy policy, which acts greedily with probability $(1-\epsilon)$, and randomly selects an action with probability ϵ.

The ϵ-greedy policy serves as an answer to the *Exploration/Exploitation-Dilemma*, by exploring the environment with probability ϵ, and exploiting current knowledge otherwise.

2.3.3 Deep Q-Network

Mnih et al. (2015) propose the Deep Q-Network algorithm (DQN), which is able to learn and play many video games better than humans. DQN is a Q-Learning algorithm that acts according to an ϵ-greedy policy. It approximates the Q-function using a deep convolutional neural network, which is able to store complex information like video game dynamics. The deep Q-network is optimized during every optimization step using stochastic gradient descent.

In order to arrive at a stable and efficient algorithm, DQN makes use of two notable modifications to basic Q-Learning: The first one, Experience Replay, stores past trajectories in a replay buffer. During every optimization step, a minibatch of transitions (s, a, r, s') is sampled from the replay buffer and used to compute the loss and gradient for the optimizer step.

The second modification is the use of a separate target Q-network. The loss that is minimized is

$$\left(R_j + \gamma \max_a \hat{q}(s', a) - q(s,a)\right)^2 \tag{2.5}$$

Instead of using the Q-network $q(s', a)$ itself as the target Q-value to be predicted, DQN uses $\hat{q}(s', a)$. $\hat{q}$ is a lagging copy of the Q-network that is updated after several optimization steps. This improves the stability of the algorithm.

2.3.4 Distributional Reinforcement Learning

The Q-function as defined in equation 2.1 is a point estimate for the expected value of total discounted return, given a state, an action, and a policy. Instead of only estimating the expected value, which is the mean of a probability distribution, one can also approximate the entire distribution $Z_\pi(s, a)$ of total discounted return.

$$Z_\pi(s, a) = p(G_t | S_t = s, A_t = a) = p\left(\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a\right) \quad (2.6)$$

$Z_\pi(s, a)$ can be modelled in multiple ways, for example as a Gaussian distribution (Azizzadenesheli et al., 2018) or as parameterized by quantiles (Dabney et al., 2018). The exact approximation procedure depends on the chosen model.

The benefit of maintaining an estimate of the entire reward distribution means the algorithm can use Thompson sampling (Thompson, 1933) in order to explore more efficiently. This amounts to selecting each action with the probability of it being the best action and has shown promising performance in some problems (Osband et al., 2013). Thompson sampling is efficient because it quickly leads to a reduction in uncertainty for promising actions, while only rarely wasting an environment interaction on unpromising actions.

However, maintaining and approximating a full estimate of the reward distribution is resource intensive, so other methods can be used to explore more efficiently. An example is to use a fixed or tuned decay schedule for the exploration rate of an ϵ-greedy strategy. The ϵ-greedy strategy prioritizes only the action currently expected to be the best one, but leaves room for exploration in the beginning, when uncertainty about action values is still high. Boltzmann sampling is another action selection strategy that prefers actions with higher Q-values, but does not solely focus on the single best action, by sampling according to the softmax distribution of their Q-values.

2.4 Reinforcement Learning for Building Demand Response

Wang and Hong (2020) list three challenges that need to be overcome before applying RL for building control: First, the training process is too timeconsuming and data demanding. Second, RL control is not secure and robust enough. And third, RL controllers need to generalize better. Consequently, research into RL for building control is rarely done in real-world situations.

Vazquez-Canteli et al. (2020) introduce CityLearn, a simulation environment for the application of RL for building energy storage control. The public CityLearn Challenges, held yearly from 2020-2022 invited numerous applications of different RL algorithms to the framework, for example Deltetto et al. (2021), Pinto et al. (2022) or Schreiber et al. (2020). Therefore, CityLearn to some extent serves as a benchmark for applying RL algorithms to building energy storage control.

Chapter 3

Methods and Approach

I evaluate the Uncertainty-Aware Deep Q-Network (UA-DQN) algorithm empirically for the task of building energy control. The experiment is realized using CityLearn to construct the reinforcement learning environment, using data from the 2022 CityLearn challenge. As a baseline, I establish a rule-based policy based on patterns in the data. Finally, I compare UA-DQN against the baseline and two versions of the DQN algorithm that differ in their action selection.

3.1 Environment

The Reinforcement Learning environment used by the experiments in this thesis is based on the 2022 CityLearn Challenge[1]. The environment provides a simulation of building energy systems, along with hourly data on electricity price, usage, solar production and weather data. The task is to control the storage and release of electricity in an electrical battery, with the objective of jointly reducing total per-building cost and carbon emissions. In contrast to the complete challenge setup, the experiments described in this thesis only use one building.

The simulation framework used by both the challenge and this thesis is CityLearn as proposed by (Vázquez-Canteli et al., 2019).

3.1.1 Data

The Dataset provided for the 2022 CityLearn challenge setup contains a year of hourly observations of a number of variables that describe the energy system of a building. At it's core, it supplies real-world per-building measurements of

[1] `https://www.aicrowd.com/challenges/neurips-2022-citylearn-challenge`

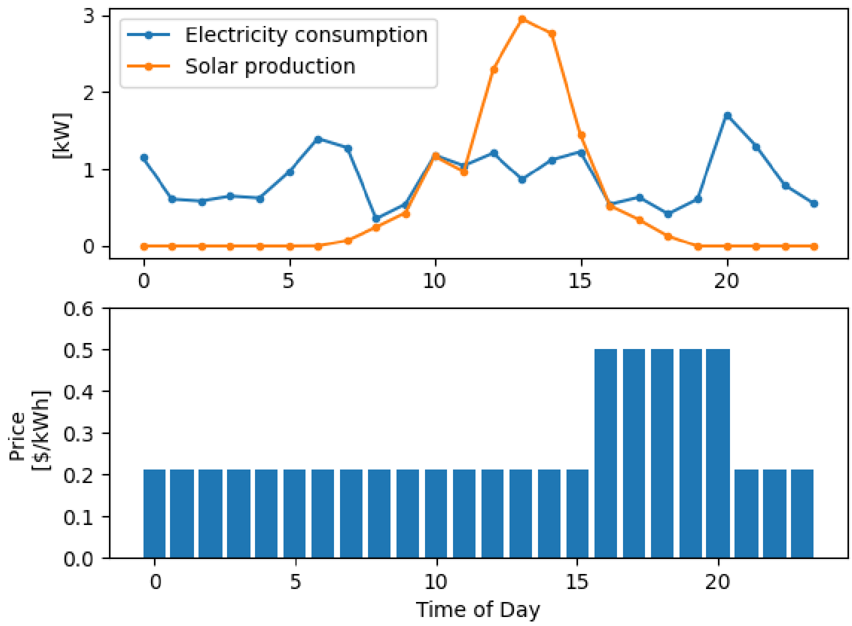

Figure 3.1: A sample day of electricity production, consumption and price for one building.

electricity use and photovoltaic solar power generation from five model buildings set up by the Electric Power Research Institute (EPRI) in Fontana, California as part of the research described in Narayanamurthy et al. (2016). Building data is coupled with weather observations (Outdoor Temperature, Relative Humidity, Diffuse and Direct Solar Radiation). Weather forecasts are also provided, which are generated by shifting ground truth observations by an offset of 6, 12 and 24 hours. The data also contains carbon intensity and price of electricity provided by the grid. Time variables included are hour of the day, day of the week and month. The source for the non-building data is unfortunately not given by the challenge organizers.

A sample day of building and price data is shown in figure 3.1. In building 1, daily solar production exceeds building electricity consumption on 72/365 days.

3.1.2 Environment Details

Observation Space

For this research, I make available a subset of the provided dimensions, given in table 3.1. Observations are dynamically normalized to zero mean and unit variance.

Table 3.1: The observed variables available in the experiments. Observations are dynamically normalized to the same scale.

Variable	Unit
Hour	1-hot encoding 0-24
Direct Solar Irradiance (predicted 6h)	W/m^2
Carbon Intensity	kg/kWh
Building Electric Load	kW
Building Solar Generation	kW
Building Battery State of Charge	kWh

Action Space

The action space provided by CityLearn is the continuous real interval $[-1, 1]$, where negative actions are an attempt to discharge, and positive actions are an attempt to charge the battery. The action is scaled in units of the battery's capacity, so -1 means an attempt to discharge the whole battery.

The actual resulting charging and discharging speeds are limited by CityLearn's energy model and depend on the battery's state of charge.

The studied Reinforcement Learning algorithms require a discrete action space. I discretize the action space into a number of discrete actions. The number of discrete actions is determined with the experiment described in section 3.3.1.

Reward Function

The reward function is designed to match the initial challenge objective as closely as possible. The per-building reward at time step t is given by

$$r_t = -\left(\frac{\text{cost}_t}{\text{cost}_{\text{no battery total}}} + \frac{\text{carbon emissions}_t}{\text{carbon emissions}_{\text{no battery total}}}\right) \cdot 8760,$$

where ()$\text{cost}_{\text{no battery total}}$) and ($\text{carbon emissions}_{\text{no battery total}}$) are the total dollar cost and carbon emissions observed over one year in a control situation where the battery is never used. The reward is always negative.

3.1.3 Implementation Details

During the 2022 CityLearn Challenge, I contributed work to the CityLearn framework. I found and proposed a fix for an implementation error that meant that the environment would recompute the entire episode history at every time step t. My fix[2] instead reuses the result of the preceding time step $t-1$, which changes the per-step complexity from $O(t^2)$ to $O(1)$, roughly leading to a 100x-speedup over the course of an episode. I also contributed to finding a bug[3] in the battery model. These efforts were rewarded with the Community Contribution Prize.

The software environment for all experiments uses Python 3.10.9, PyTorch 1.13.0, OpenAI gym 0.24.1, and CityLearn 1.3.6.

The hardware used was a 2020 MacBook Air M1 for the discretization pre-experiment. Tuning and subsequent evaluation runs were scheduled on a Slurm cluster, using CUDA on a single Nvidia GTX 2080 GPU per run. The Slurm job scheduling file is included in the attached code repository.

3.2 Algorithm

3.2.1 Uncertainty-Aware Deep Q-Network

In RL, it is important to distinguish between epistemic uncertainty, which is caused by limited data, and aleatoric uncertainty, which is caused by stochasticity in the environment (Hüllermeier and Waegeman, 2021). Efficient exploration aims to reduce epistemic uncertainty for promising actions, but can't reduce aleatoric uncertainty. Nikolov et al. (2022) propose to estimate aleatoric uncertainty using the variance of the estimated reward distribution. However, Chua et al. (2018) highlight that this approach fails for novel, out-of-distribution data, when epistemic uncertainty is high.

The UA-DQN algorithm proposed by Clements et al. (2020) learns a distribution of expected rewards, and simultaneously provides estimates for epistemic and aleatoric uncertainties. These uncertainties are used to adjust the expected value and variance for each action, before using Thompson sampling to select an action. This leads to an algorithm that can efficiently explore. Depending on its hyperparameters, the algorithm can be more or less risk-averse.

Efficient exploration and risk-aware strategies are most important when the costs and risks associated with an environment interaction are high, as they are in a real-world energy setting. Therefore, I test the performance of the UA-DQN algorithm on the battery control environment, replicating the experiments described in Clements et al. (2020).

[2] `https://github.com/intelligent-environments-lab/CityLearn/pull/23`
[3] `https://github.com/intelligent-environments-lab/CityLearn/issues/37`

3.2.2 DQN

In order to study the benefit of an explicit uncertainty treatment, I evaluate the performance of two variants of the standard DQN algorithm. Like in Clements et al. (2020), one DQN variant acts according to an ϵ-greedy policy, which linearly decays ϵ from 1.0 to 0.02 in the first 1000 steps. The second variant, referred to throughout the thesis as "DQN-Softmax" uses Boltzmann sampling on the estimated Q-values.

3.2.3 Implementation Details

The implementation for all three Reinforcement Learning agents is based on the implementation provided with Clements et al. (2020) with minimal changes. All neural networks have two hidden layers with 128 neurons each and ReLU activation functions. They are randomly initialized with an orthogonal matrix as described by Saxe et al. (2014), with a weight scale factor of $\sqrt{2}$.

All algorithms use the Adam optimizer, with tuned ϵ and learning rate parameters. All algorithms use experience replay and a periodically updated target Q-network.

UA-DQN is set to be risk-neutral, with the aleatoric factor fixed to 0 and the epistemic factor fixed to 1. Loss functions are Mean Squared Error for DQN, and Quantile Huber Loss for UA-DQN.

3.2.4 Baseline Rule-Based-Controller

In order to be able to evaluate the performance of the Reinforcement Learning agent, I establish a rule-based controller as baseline. Using insights gained from exploratory data analysis, I construct a simple control policy with the goal of minimizing dollar cost.

The basis of the policy is that prices vary predictably. Throughout every day, electricity price is at one of two levels. From 16:00 to 20:00, it is substantially higher than during the rest of the day. In parts of the year, the price level also varies between different days of the week, but the daily pattern still applies. Since battery energy losses are very low, it is therefore worth it to buy and store electricity while it's cheap, and avoid having to buy expensive electricity in the afternoon.

In addition to the electrical grid, the agent also has access to a free, but less predictable source of electricity: solar power. The environment does not allow the agent to sell electricity for a profit. This means any produced solar electricity should either be directly used or stored, since any excess is simply discarded. Directly using solar electricity is more efficient than storing and releasing it at a later time. Putting these basic ideas together, I arrive at a policy that first uses available solar electricity to meet demand. It aims to always store excess solar electricity, and additionally buys just enough cheap electricity in order to fill up the battery for the more expensive afternoon.

Because the policy has to make a decision based on the past hour's solar production and electricity consumption, it simply assumes there's no change and tries to store the amount of excess electricity observed before.

This strategy leaves open one question: When exactly should the battery be charged with electricity from the grid? If it is filled too early, the agent is not able to store excess solar electricity generated during the day. Therefore, the battery is charged as late as possible. However, if battery charging starts too late, there is not enough time left to fully charge the battery. Lastly, if the battery has remaining charge after the period of high prices, the leftover charge is used as needed, ensuring the battery is free in the morning to store any excess solar power. The final policy is described in algorithm 1.

Algorithm 1 The Rule-Based Controller's Policy

```
a ← (solar − load)/6.4          ▷ Difference scaled to units of battery capacity
if 11 ≤ hour ≤ 15 then
    a ← max(0.24, a)            ▷ Slowly charge battery
end if
Ensure: −1 ≤ a ≤ +1
return a
```

This hand-engineered policy is not perfectly optimized. There are certain insights it does not make use of. It does not directly minimize carbon emissions, though the overall reduced demand for grid electricity leads to a reduction in emissions. The policy also does not incorporate the change in price between weekdays. There are some days on which there is so little excess demand during the day that it would not be worth charging the battery. Finally, the policy does not make use of weather forecasts, which predict solar electricity production.

Overall, the policy is a useful benchmark of expert human performance.

3.3 Experiment

3.3.1 Discretization

UA-DQN and DQN require a discrete action space. In CityLearn, the action is a continuous number between -1 (releasing energy) and +1(storing energy). Therefore, I subdivide the continuous action space into multiple discrete actions. The goal of this pre-experiment is to determine a suitable resolution

Table 3.2: Hyperparameters, their tuning ranges or non-tuned values.

Hyperparameter	Tuning Range / Value
Learning Rate	$(0.1, 0.07, 0.03, 0.01, 0.007, \ldots, 0.00001)$
Batch Size	$(1, 2, 4, 8, \ldots, 256)$
Adam's ϵ	$(1 \times 10^{-1}, 1 \times 10^{-2}, \ldots, 1 \times 10^{-9})$
Target Network Update Frequency	$(4, 8, 16, 32)$
Replay Buffer Size	10,000
Discount Rate γ	0.99
Initialization Weight scale	$\sqrt{2}$
ϵ-greedy DQN: ϵ	Decay from 1 to 0.02 over 1,000 steps
UA-DQN: Quantile Huber Loss κ	10
UA-DQN: aleatoric factor	0
UA-DQN: epistemic factor	1

of subdivision of the continuous action space. A larger discrete action space means the algorithm learns slower, but a smaller discrete action space means the algorithm can't act as precisely, capping the possible performance. The goal therefore is to find the smallest number of subdivisions that does not produce a significant drop in performance.

In order to measure the importance of different subdivisions, I run the baseline rule-based policy on versions of the environment with differently discretized action spaces. The environment uses the entire 2022 CityLearn challenge public dataset of 5 buildings and one year.

I measure the performance as calculated by CityLearn, both cost and carbon metrics. The performance on the discretized action spaces is compared with the policy's performance on the continuous action space.

3.3.2 Hyperparameter Tuning

All tested Deep Reinforcement Learning agents and the optimizer, Adam, expose hyperparameters that need to be tuned for optimal performance. Adam's tuned hyperparameters are the learning rate and the parameter ϵ. I also tuned the batch size and the update frequency of the target networks. All other hyperparameters I set to default values as noted in table 3.2.

The process of tuning was to randomly sample 200 combinations of hyperparameters from the sets given in table 3.2 for each DQN variant, and 100 for UA-DQN. The DQN variants ran for 100 episodes, UA-DQN ran for 50 episodes. All runs used the same random seed. Data used for this experiment was the whole year of building 1.

The performance measure for this experiment was the collected total reward in the last episode. For each algorithm, I select the single best performing combination of hyperparameters from the tuning runs for further study.

3.3.3 Comparison of Tuned Algorithms

Following the analysis of Clements et al. (2020), I measure the fraction of non-greedy actions for each algorithm during a training run of 50 episodes. A non-greedy action is an action that does not correspond to the algorithm's highest expectation for expected total reward. Instead, it was taken in order to explore the environment rather than exploit current knowledge. This illustrates the difference in how the algorithms balance exploitation and exploration as they learn.

Finally, I repeat the previous experiment with 10 different random seeds in order to be able to measure the repeatability and resource demands of each algorithm. I evaluate each algorithm by its mean performance compared to the baseline and by its variance in final performance. In order to evaluate the difference in resource use between the algorithms, I also measure the average wall clock time taken per training episode.

Chapter 4

Results

4.1 Discretization

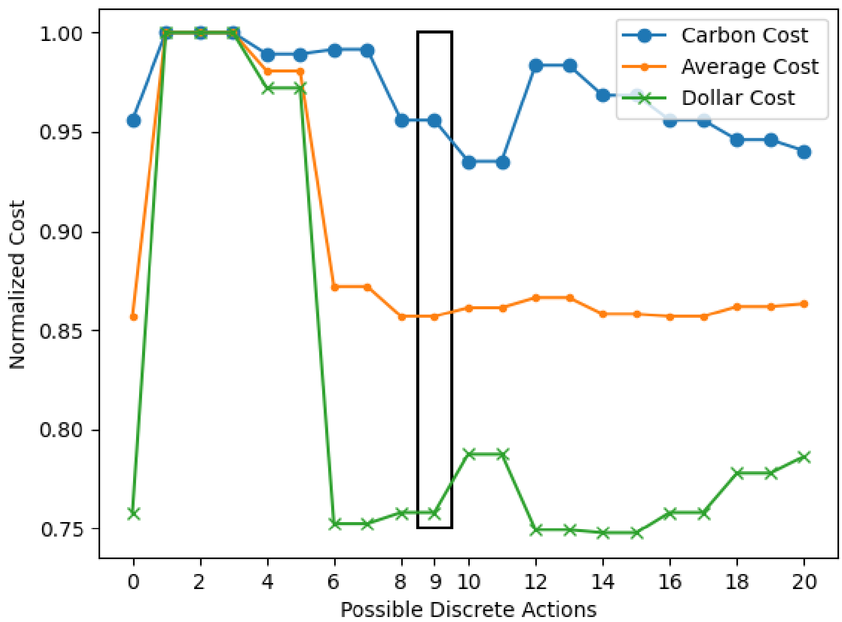

Figure 4.1: This graph shows the effect of the discretization resolution on the rule-based control algorithm. The coarsest resolution that does not significantly impact performance and includes the zero action is 9. In this graph, 0 possible actions means no discretization is applied.

The results from the pre-experiment on the effect of discretization are shown in figure 4.1. The rule-based agent performs comparably to the continuous case when discretization resolution is high, and worse on coarse resolutions.

The coarsest resolution that performs similarly well as the continuous case is the subdivision into 8 or 9 possible actions.

4.2 Hyperparameter Tuning

All 200 runs of ϵ-greedy DQN completed successfully. All 10 UA-DQN runs with a batch size of 1 failed. 18/200 runs of DQN-Softmax failed, all of which share a batch size of 8. However, 17 other runs with a batch size of 8 completed successfully.

Table 4.1 shows the results of the tuning process for each algorithm. For each algorithm, the learning rate had the most significant correlation with final performance, followed by the batch size and the target network update frequency. The ϵ parameter of the optimizer Adam showed a large effect only for ϵ-greedy DQN.

Figure 4.2 shows a histogram of the final performance of all successful tuning runs. 52% of all UA-DQN runs perform better than the idle policy, 40.5% of all DQN-Softmax runs perform better than the idle policy, and 37% of e-greedy DQN runs perform better than the idle policy.

Both DQN algorithms show a peak at around -5000 total reward in the final episode, which corresponds to strategies that are close to the idle policy.

Table 4.1: This Table shows the correlation between tuned hyperparameters and algorithm performance for successful runs.

Algorithm	Hyperparameter	Value for Best Run	Correlation with final score
UA-DQN	Learning Rate	3e-4	**-0.47**
	Batch Size	128	0.28
	Adam's ϵ	1e-07	-0.03
	Update Frequency of Target Network	4	-0.11
DQN-Softmax	Learning Rate	3e-4	**-0.36**
	Batch Size	128	-0.18
	Adam's ϵ	1e-05	-0.02
	Update Frequency of Target Network	4	0.16
ϵ-greedy DQN	Learning Rate	7e-05	**-0.33**
	Batch Size	4	-0.19
	Adam's ϵ	1e-08	0.10
	Update Frequency of Target Network	16	0.14

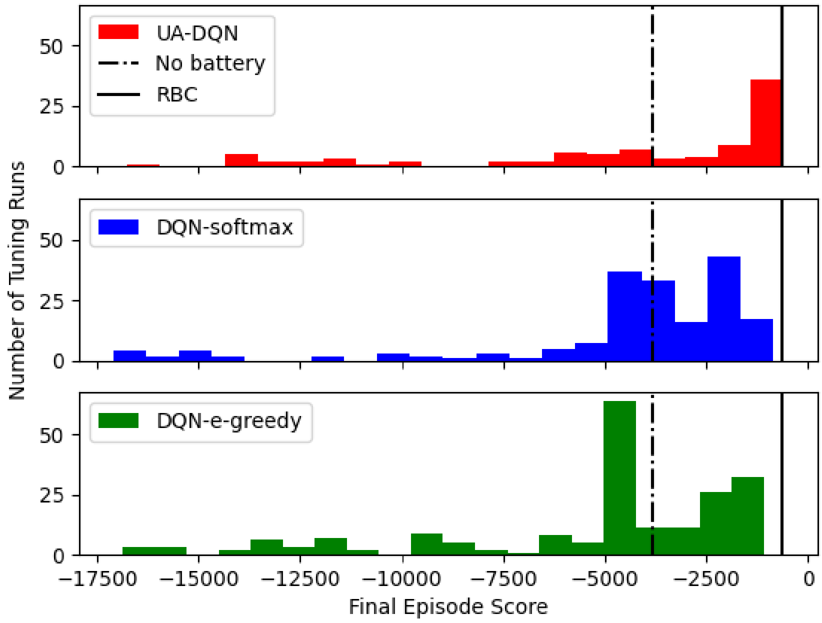

Figure 4.2: This histogram shows the final episode reward of all successful tuning runs. Every run represents one choice for the hyperparameters. The performance of the rule-based controller and the idle policy are shown for reference.

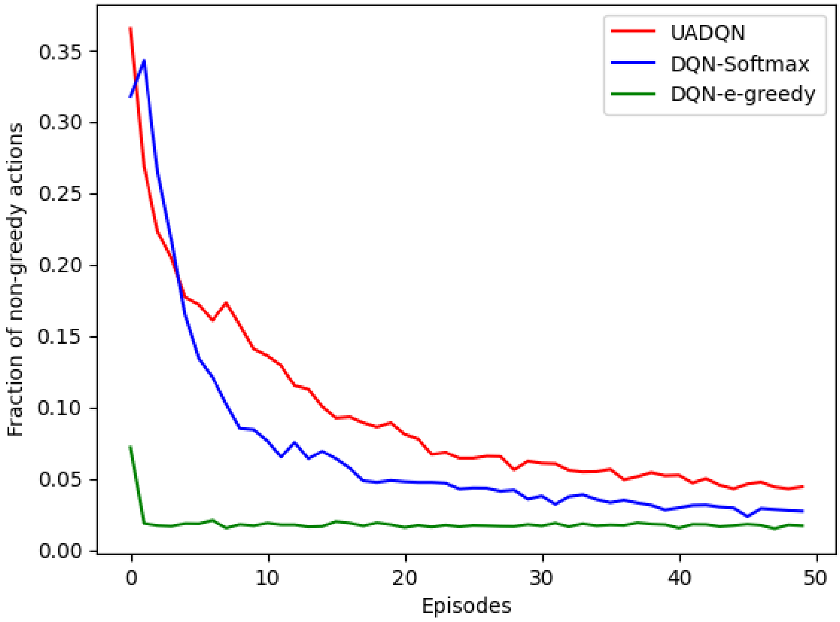

Figure 4.3: This graphic shows the exploration rate of tuned algorithms throughout the training process. It highlights the different action selection strategies employed by the different algorithms. UA-DQN keeps exploring more than the other strategies.

4.3 Comparison of Tuned Algorithms

To illustrate the difference in exploration between the algorithms, figure 4.3 shows the fraction of selected non-greedy actions per episode. All algorithms start out exploring more and then gradually decrease their exploration rate. DQN-Softmax and UA-DQN explore more than ϵ-greedy DQN, which quickly reaches an exploration rate of $\epsilon = 0.02$. UA-DQN keeps exploring more than the other algorithms.

Figure 4.4 shows the episode rewards of the selected hyperparameters for each algorithm during training. Tuned UA-DQN converges faster than the other algorithms, and it converges to a better mean performance, as shown in table 4.3. The hand-engineered rule-based agent outperforms the tuned reinforcement learning algorithms on both emission and cost metrics.

When repeated with 10 different seeds, a single run of DQN-Softmax failed, compared to no failures from the other algorithms.

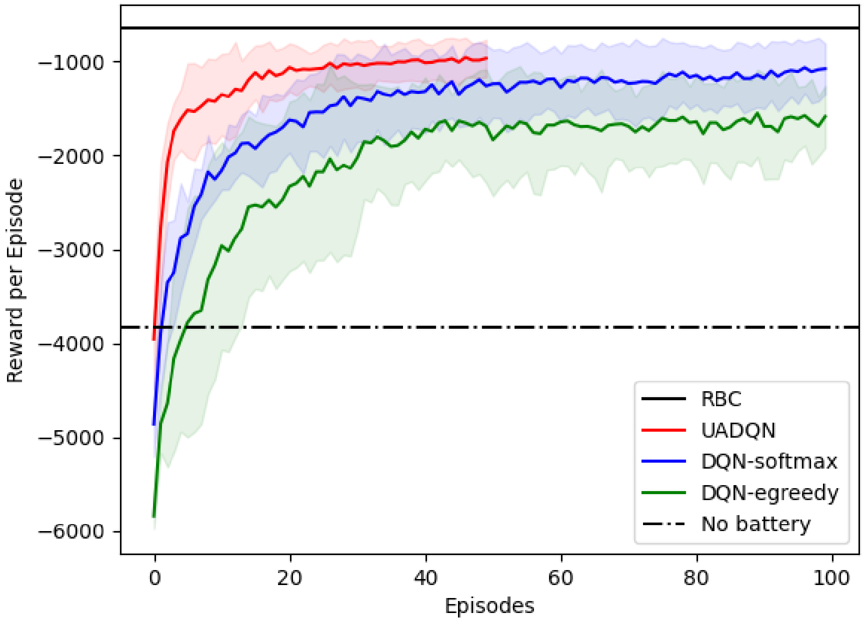

Figure 4.4: This graphic shows the performance during training of the three tuned algorithms. The shaded area shows the standard deviation over 10 runs with different random seeds. Tuned UA-DQN converges after fewer episodes than either DQN variant. UA-DQN was only trained for 50 episodes due to the larger resource requirements of the algorithm.

Table 4.2: This table shows the mean time per training episode for each of the tuned algorithms.

Agent	Mean Time per Training Episode
UA-DQN	95.3s
DQN-Softmax	27.2s
ϵ-greedy DQN	20.7s

Table 4.3: This table shows the mean performance of tuned algorithms when evaluated using their respective action selection policy for one episode on Building 1. I also give the standard deviation of the combined average.

Agent	Dollar Cost	Carbon Emission	Average
Control (Idle)	1	1	**1**
ϵ-greedy DQN	0.83	0.94	$\mu = \mathbf{0.88}, SD = 0.017$
DQN-Softmax	0.83	0.93	$\mu = \mathbf{0.88}, SD = 0.019$
UA-DQN	0.82	0.91	$\mu = \mathbf{0.87}, SD = 0.010$
Discrete Rule-Based	0.80	0.88	**0.84**

Chapter 5

Discussion

Through the experiments described in the previous chapters, I evaluate the Uncertainty-Aware Deep Q Network algorithm on a custom building-scale energy-management environment based on the 2022 CityLearn challenge. Compared to two versions of the simpler DQN algorithm, UA-DQN requires fewer interactions with the environment to arrive at a good performance. However, it does not reliably exceed the performance of a baseline rule-based control policy. In this chapter, I discuss the collected evidence and limitations of the experiment and point out several possible ways of further improving the algorithm's performance.

5.1 Preparation

The baseline rule-based policy was manually derived from simple patterns observed in the data. Its performance is compared to the performance of an idle policy, which does not make use the battery at all. It enables substantial savings in electricity cost and carbon emissions. Compared to the idle policy, more of the generated solar electricity is used. Therefore, overall, less electricity is purchased from the grid. The policy also shifted the time of electricity purchase to an earlier time of day. Especially during peak hours, when both price and carbon intensity are high, the policy achieves a substantial reduction in building electricity demand. The policy does dynamically adapt to observed solar generation and electricity use, but it can not predict the amount of excess solar production to be stored. As an approximation, it assumes no change from the hour before in the amount of excess solar production. This is an obvious limitation of the policy.

Overall, the rule-based policy can serve as a useful baseline for evaluating the performance of more complex algorithms. It is a simple algorithm that captures a lot of the storage potential of the system, but is not finely optimized on minute details.

In order to determine the simplest discrete action space for the DQN variants that would allow them to perform well, I perform a pre-experiment on the rule-based policy. The experiment tests the performance of the rule-based policy on different resolutions of discretized action space against its performance on the continuous action space.

Figure 4.1 shows a large dependency of the rule-based policy on resolution of the action space. As expected, a higher discretization resolution approaches the continuous case and leads to better performance. However, the policy's performance is particularly impacted because it relies on a fixed number as minimum charging action of 0.24 in the hours before the high-price period starts in order to ensure the battery is fully charged. Depending on the resolution of discretization, this number is mapped to a less useful fixed value. Therefore, the chosen resolution of 9 discrete actions favors policies that work very similarly to the rule-based strategy.

5.2 RL Algorithms

After establishing the baseline and environment details, I identify and tune important hyperparameters of UA-DQN and two versions of DQN for best performance. I then rerun the tuned algorithms with multiple random seeds to get an estimate of their reliability.

The results of the tuning runs, presented in figure 4.2, show that the performance of all algorithms depends on hyperparameter choice. Comparing the algorithms, UA-DQN performs better for a wider range of hyperparameter choices. For each algorithm, the most important hyperparameters are reported in table 4.1. Across all algorithms, the learning rate is the most important hyperparameter as measured by correlation with the final score. On average, a smaller learning rate leads to a better performance. This is mainly due to the fact that a high learning rate destabilizes the algorithm. Adam's ϵ did not have a large effect on either algorithm.

Some tuning runs failed. All UA-DQN runs with a batch size of 1 failed because the quantile Huber loss function needs a batch size of at least 2, but for correct hyperparameters the algorithm proved robust. The DQN-Softmax algorithm failed to converge on some runs with a batch size of 8. Even the tuned variant only completed 9/10 runs. This suggests some inherent instability of the algorithm.

As described before, the promise of UA-DQN is that it explores more efficiently than simpler algorithms. Figure 4.3 shows the fraction of non-greedy actions taken by the tuned algorithms. A non-greedy action is an action that was taken in order to explore the environment rather than exploit current knowledge. In first few episodes, DQN-Softmax explores more than the other algorithms. UA-DQN explores more throughout the rest of the training run than the other strategies. ϵ-greedy DQN explores much less than the other strategies. A more sophisticated and tuned schedule for the exploration rate ϵ could improve the algorithm's performance at the added cost of requiring more tuning. Figure 4.4 shows that UA-DQN improves its performance much faster than the other algorithms. Especially in the first 5 episodes, when UA-DQN explores less than DQN-Softmax, UA-DQN performance improves much faster. The reason for this could be either that tuned UA-DQN takes more useful exploratory actions, or that it integrates observed information more effectively, or a combination of both.

The used tuning method favors risky hyperparameter choices. Because every combination of hyperparameters is tried only once, accepting the single best tuning run probably means accepting an outlier performance for this combination of hyperparameters. For example, the best performing runs of UA-DQN and DQN-Softmax share a low target network update frequency, which causes both faster learning and instability. Similarly, the best performing run of ϵ-greedy DQN uses a small batch size of 4. Therefore, the hyperparameters selected by the tuning process are biased towards being less reliable. This effect might have further helped UA-DQN, which was tuned on half as many runs as the other algorithms, so there was less opportunity for risky hyperparameters to outperform more robust choices. However, even with fewer tuning runs, the best UA-DQN tuning run significantly outperformed the DQN variants. A different tuning procedure might have favored more reliable algorithms with a better mean performance. One further possible source of bias is the networks' initialization, which was the same for all algorithms during tuning.

In summary, the experiment shows the potential of UA-DQN, which is able to take more useful exploratory actions. The experiment suffers from multiple potential sources of bias, and a tuning procedure that favors unstable hyperparameters. UA-DQN approaches the performance of the rule-based baseline and learns from fewer environment interactions than simpler DQN variants, but has much larger resource demands (see table 4.2).

5.3 General Discussion

The custom environment based on the 2022 CityLearn challenge presents a planning and control task based on real-world data. Some of the experiment's findings can be expected to generalize to real-world applications of the algorithms. Particularly, the environment requires the agent to learn patterns in electricity price and carbon intensity, and domestic electricity usage and production. The strategy task of when to use electricity storage is modelled well by CityLearn. Success in this task depends on long-term, aggregate data and patterns on the scale of multiple hours. In contrast, a part of the rule-based policy's aim is to try to store precisely the amount of excess electricity generated. This is nearly impossible to do in CityLearn, but easy to implement in a real-world building control system. This limits the performance of a good long-term plan, as evaluated by CityLearn.

A real-world task would have access to more data than the model environment used in the experiments. This would favor machine learning algorithms. The environment used in the experiments only provides a subset of CityLearn's data as observed features. For example, features included only one weather forecast variable, no information on day of the week or month, and no price forecasts. Real-world weather forecasts are uncertain, which requires a different strategy for acting under uncertainty. In a real-world application, a smart home system might also have access to occupant behavioral data, like knowing if someone is home or not, which can lead to significantly better plans. While the building data stems from only one building in a particular climate zone, the algorithms' effectiveness depends mostly on the daily patterns of pricing. More complex or dynamic pricing models might require more sophisticated planning algorithms. In fact, coordination strategies such as dynamic pricing are another goal not addressed by this environment. A real-world system could likely not only buy, but also sell electricity to and from the grid, providing flexibility not only to the building.

The CityLearn framework is useful for developing experiments that test the application of RL planning algorithms to building energy control tasks. It's an efficient environment that models important parts of a building and can incorporate real-world data. However, in the current version, it does not serve as a realistic model for the full task. Applying RL for building energy control is an area of ongoing research, and Nweye et al. (2022) name a number of challenges to be overcome when applying RL to grid-integrated buildings.

The experiment described in this thesis shows that UA-DQN can explore a simple environment more efficiently than other RL algorithms. This provides further evidence for the usefulness of uncertainty-aware RL methods when acting in high-stakes environments. However, in this experiment, UA-DQN did not outperform the rule-based policy. A real-life building energy demand response task might be more complex, which might favor a machine learning approach.

Chapter 6

Conclusion and Outlook

UA-DQN is a Reinforcement Learning algorithm based on the DQN algorithm. It learns the reward distribution and provides estimates for both epistemic and aleatoric uncertainty. This allows it to explore more efficiently, and act according to a risk-aware strategy. In this thesis, I apply the UA-DQN algorithm to a custom task for building energy management implemented in CityLearn. I find that tuned UA-DQN outperforms two variants of DQN, reaching comparable performance with fewer environment interactions. However, this added efficiency comes at a cost: UA-DQN has roughly three times as much computational cost as the simpler algorithms. None of the tuned algorithms outperform an optimized rule-based control policy.

Like other algorithms that explicitly represent uncertainty in addition to point estimates, UA-DQN benefits from this additional information. Application of UA-DQN and similar algorithms should be considered in problems where additional observations are expensive or bad decisions are costly. Additionally, UA-DQN might perform particularly well on non-stationary problems, when local weather patterns or occupant behavior changes. Future work can also generalize the method behind UA-DQN to a continuous action space.

CityLearn is a suitable framework to model a building energy management task for Reinforcement Learning, but it remains a challenge to transfer insights into real-world applications because of modeling and data decisions. Future CityLearn challenges should play to its strengths and focus on longer-term planning rather than short-term control.

In the context of automated Demand Response by building energy systems, there remain many exciting challenges for Uncertainty-Aware Reinforcement Learning. One interesting line of work is multiagent learning and coordination mechanisms, as they will allow direct optimization for grid-scale goals. Future work on uncertainty-aware Reinforcement Learning methods for building control should seek to incorporate uncertain data like weather forecasts or models of occupant behavior.

Bibliography

Kamyar Azizzadenesheli, Emma Brunskill, and Animashree Anandkumar. Efficient Exploration Through Bayesian Deep Q-Networks. In *2018 Information Theory and Applications Workshop (ITA)*, pages 1–9, February 2018. doi: 10.1109/ITA.2018.8503252. 2.3.4

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. 3.2.1

William R. Clements, Bastien Van Delft, Benoît-Marie Robaglia, Reda Bahi Slaoui, and Sébastien Toth. Estimating Risk and Uncertainty in Deep Reinforcement Learning. *arXiv preprint arXiv:1905.09638*, September 2020. doi: 10.48550/arXiv.1905.09638. 1, 3.2.1, 3.2.2, 3.2.3, 3.3.3

Will Dabney, Mark Rowland, Marc Bellemare, and Rémi Munos. Distributional Reinforcement Learning With Quantile Regression. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), April 2018. ISSN 2374-3468. doi: 10.1609/aaai.v32i1.11791. 2.3.4

Davide Deltetto, Davide Coraci, Giuseppe Pinto, Marco Savino Piscitelli, and Alfonso Capozzoli. Exploring the Potentialities of Deep Reinforcement Learning for Incentive-Based Demand Response in a Cluster of Small Commercial Buildings. *Energies*, 14(10):2933, January 2021. ISSN 1996-1073. doi: 10.3390/en14102933. 2.4

Marcos Gimeno-Gutiérrez and Roberto Lacal-Arántegui. Assessment of the European potential for pumped hydropower energy storage based on two existing reservoirs. *Renewable Energy*, 75:856–868, March 2015. ISSN 0960-1481. doi: 10.1016/j.renene.2014.10.068. 1, 2.1

Eyke Hüllermeier and Willem Waegeman. Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Machine Learning*, 110(3):457–506, March 2021. ISSN 1573-0565. doi: 10.1007/s10994-021-05946-3. 3.2.1

P. Khajavi, H. Abniki, and A. B. Arani. The role of incentive based Demand Response programs in smart grid. In *2011 10th International Conference on Environment and Electrical Engineering*, pages 1–4, May 2011. doi: 10.1109/EEEIC.2011.5874702. 2.2

Kourosh Latifi, Artur Kopitca, and Quan Zhou. Model-Free Control for Dynamic-Field Acoustic Manipulation Using Reinforcement Learning. *IEEE Access*, 8:20597–20606, 2020. ISSN 2169-3536. doi: 10.1109/ACCESS.2020.2969277. 1

Han Li, Zhe Wang, Tianzhen Hong, and Mary Ann Piette. Energy flexibility of residential buildings: A systematic review of characterization and quantification methods and applications. *Advances in Applied Energy*, 3:100054, August 2021. ISSN 2666-7924. doi: 10.1016/j.adapen.2021.100054. 2.2

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015. ISSN 1476-4687. doi: 10.1038/nature14236. 1, 2.3.3

Ram Narayanamurthy, Rachna Handa, Nick Tumilowicz, C R Herro, and Sunil Shah. Grid Integration of Zero Net Energy Communities. *ACEEE Summer Study Energy Effic. Build*, 2016. 3.1.1

Nikolay Nikolov, Johannes Kirschner, Felix Berkenkamp, and Andreas Krause. Information-Directed Exploration for Deep Reinforcement Learning. In *International Conference on Learning Representations*, February 2022. 3.2.1

Kingsley Nweye, Bo Liu, Peter Stone, and Zoltan Nagy. Real-world challenges for multi-agent reinforcement learning in grid-interactive buildings. *Energy and AI*, 10:100202, November 2022. ISSN 2666-5468. doi: 10.1016/j.egyai.2022.100202. 5.3

Ian Osband, Daniel Russo, and Benjamin Van Roy. (More) Efficient Reinforcement Learning via Posterior Sampling. In *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013. 2.3.4

Giuseppe Pinto, Marco Savino Piscitelli, José Ramón Vázquez-Canteli, Zoltán Nagy, and Alfonso Capozzoli. Coordinated energy management for a cluster of buildings through deep reinforcement learning. *Energy*, 229:120725, August 2021. ISSN 0360-5442. doi: 10.1016/j.energy.2021.120725. 1

Giuseppe Pinto, Anjukan Kathirgamanathan, Eleni Mangina, Donal P. Finn, and Alfonso Capozzoli. Enhancing energy management in grid-interactive buildings: A comparison among cooperative and coordinated architectures. *Applied Energy*, 310:118497, March 2022. ISSN 0306-2619. doi: 10.1016/j.apenergy.2021.118497. 2.4

Hans-O. Pörtner, Debra C. Roberts, Helen Adams, Carolina Adler, Paulina Aldunce, Elham Ali, Rawshan Ara Begum, Richard Betts, Rachel Bezner Kerr, and Robbert Biesbroek. *Climate Change 2022: Impacts, Adaptation and Vulnerability.* IPCC Geneva, Switzerland:, 2022. 1

Hannah Ritchie, Max Roser, and Pablo Rosado. CO_2 and Greenhouse Gas Emissions. *Our World in Data*, May 2020. 1

Hannah Ritchie, Max Roser, and Pablo Rosado. Energy. *Our World in Data*, October 2022. 2.1

Andrew M. Saxe, James L. McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks, February 2014. 3.2.3

Thomas Schreiber, Sören Eschweiler, Marc Baranski, and Dirk Müller. Application of two promising Reinforcement Learning algorithms for load shifting in a cooling supply system. *Energy and Buildings*, 229:110490, December 2020. ISSN 0378-7788. doi: 10.1016/j.enbuild.2020.110490. 1, 2.4

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction.* Adaptive Computation and Machine Learning Series. The MIT Press, Cambridge, Massachusetts, second edition edition, 2018. ISBN 978-0-262-03924-6. 2.3.1

William R. Thompson. On the Likelihood that One Unknown Probability Exceeds Another in View of the Evidence of Two Samples. *Biometrika*, 25 (3/4):285–294, 1933. ISSN 0006-3444. doi: 10.2307/2332286. 2.3.4

José Vázquez-Canteli, Jérôme Kämpf, Gregor Henze, and Zoltán Nagy. CityLearn v1.0: An OpenAI Gym Environment for Demand Response with Deep Reinforcement Learning. *Proceedings of the 6th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, 2019. doi: 10.1145/3360322.3360998. 3.1

Jose R. Vazquez-Canteli, Sourav Dey, Gregor Henze, and Zoltan Nagy. CityLearn: Standardizing Research in Multi-Agent Reinforcement Learning for Demand Response and Urban Energy Management. *arXiv:2012.10504 [cs]*, December 2020. 1, 2.4

Zhe Wang and Tianzhen Hong. Reinforcement learning for building controls: The opportunities and challenges. *Applied Energy*, 269:115036, July 2020. ISSN 0306-2619. doi: 10.1016/j.apenergy.2020.115036. 1, 2.4

Dongying Zhang, Huiting Zhang, Xu Zhang, Xiaoyu Li, Kaiqi Ren, Yongxu Zhang, and Yun Guo. Research on AGC Performance During Wind Power Ramping Based on Deep Reinforcement Learning. *IEEE Access*, 8:107409–107418, 2020. ISSN 2169-3536. doi: 10.1109/ACCESS.2020.3000784. 1

Selbstständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Masterarbeit selbständig und nur mit den angegebenen Hilfsmitteln angefertigt habe und dass alle Stellen, die dem Wortlaut oder dem Sinne nach anderen Werken entnommen sind, durch Angaben von Quellen als Entlehnung kenntlich gemacht worden sind. Diese Masterarbeit wurde in gleicher oder ähnlicher Form in keinem anderen Studiengang als Prüfungsleistung vorgelegt.

Ort, Datum Unterschrift