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

Write here your abstract.

Zusammenfassung

Bei einer englischen Masterarbeit muss zusätzlich eine deutsche Zusammenfassung verfasst werden.

Acknowledgements

Write here your acknowledgements.

Contents

Chapter 1

Introduction

Start with a comprehensive introduction about the questions of your thesis. 1-2 pages:

When proofreading: Check that all terms and abbreviations are introduced here

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 ... governments have committed to an ambitious goal of drastically reducing carbon emissions to keep global warming from increasing beyond 2°C, compared to ... The latest IPCCC report urges governments to take stronger actions, or their previous commitment will not be reached. 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.

cite

cite

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 from today's...%, the reliability of the electricity supply goes down.

cite

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 market constructed on top of the physical layer. As the share of renewable power increases, fossil-fuelled plants remain the only market participants that

talk about reacting to less reliability in renew-

can flexibly react to changes in demand. 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. However, consumer electricity demand can flexibly react to changes in supply, a scheme called Demand Response. Grid-scale storage in Europe mainly consists of hydropower, which has been installed where mountainous geography allows, and capacity has reached its natural limit. More expensive battery-powered storage facilities are slowly being built, but are largely not cost-efficient in the current economic setting.

cite

In this thesis, I focus on an opportunity to make consumer electricity demand more flexible.

Buildings require energy mainly for heating and cooling the air and the water supply. Today, they are responsible for ...% of total energy demand. On the other hand, buildings often contribute to electricity production through photovoltaic panels. New buildings often come with a battery, which enables them to more efficiently use their solar electricity.

cite

Building's electricity consumption is already largely automated and is therefore a prime candidate for automated demand response.

mention incentive-based human DR

The Reinforcement Learning family of control algorithms has successfully been applied for control of the battery of simulated buildings. . A popular simulation framework for this purpose is CityLearn. In this thesis, I set out to test Uncertainty-Aware Deep Q-Networks on CityLearn. UA-DQN is an uncertainty-aware adaptation of Deep Q-Learning.

The algorithm's better 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.

talk about results!

These aspects are important for a demand response algorithm, which provides flexibility to the grid while reliably meeting building demands for energy.

list other automated control algorithms and their goals

This thesis is structured as follows: In the following chapter, 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.

cite algorithms

cite CityLearn

cite algorithm

cite DQN

Things to add in introduction: - Einordnung in die bestehende Literatur. Was ist an meinem Ansatz anders als an bisherigen Ansätzen? - Little bit of Results (abstrakt success vs failure) (technical, numbers)
Terms to check: - Demand Response vs. Demand Side Management - Reinforcement Learning - Electrical Grid

Chapter 2

Background and related work

Give all the necessary background that is needed to justify and understand the problem and the approach. - comment on employed hardware and software - describe methods and techniques that build the basis of your work - review related work(!) - roughly 1/3 of thesis

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.

Another aspect of the energy transition is 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

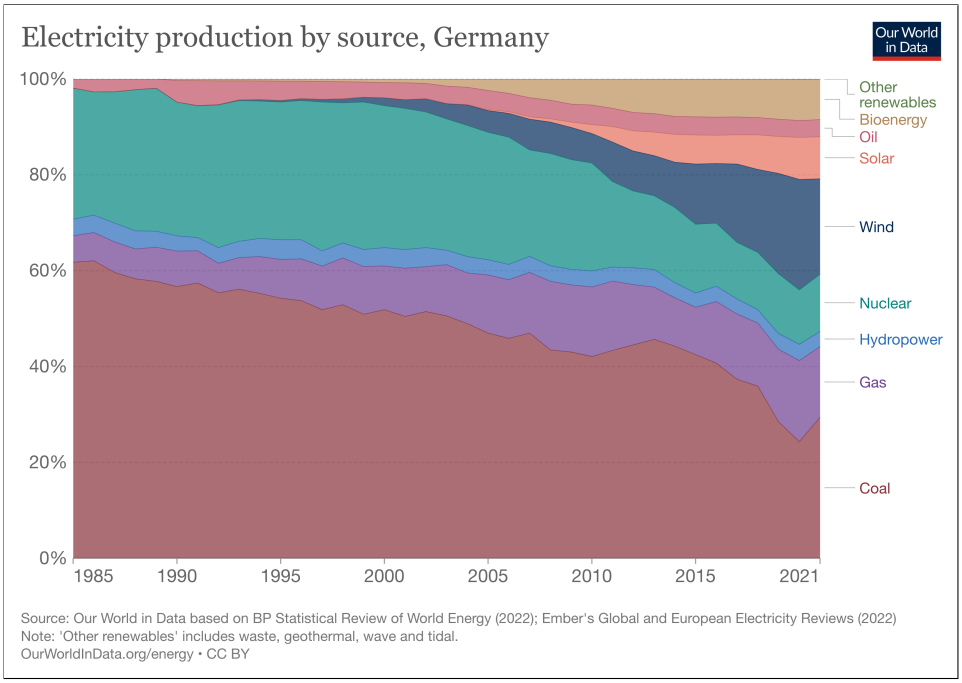


Figure 2.1: The share of renewable electricity has risen significantly from 1985 to 2021, while nuclear and fossil fuels have declined.

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 demand. 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.

maybe give specific number

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 depends almost entirely on geography. There is almost no potential to develop more hydropower storage.

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, this method is also costly, and any produced hydrogen is probably worth more as a crucial chemical than as pure electricity

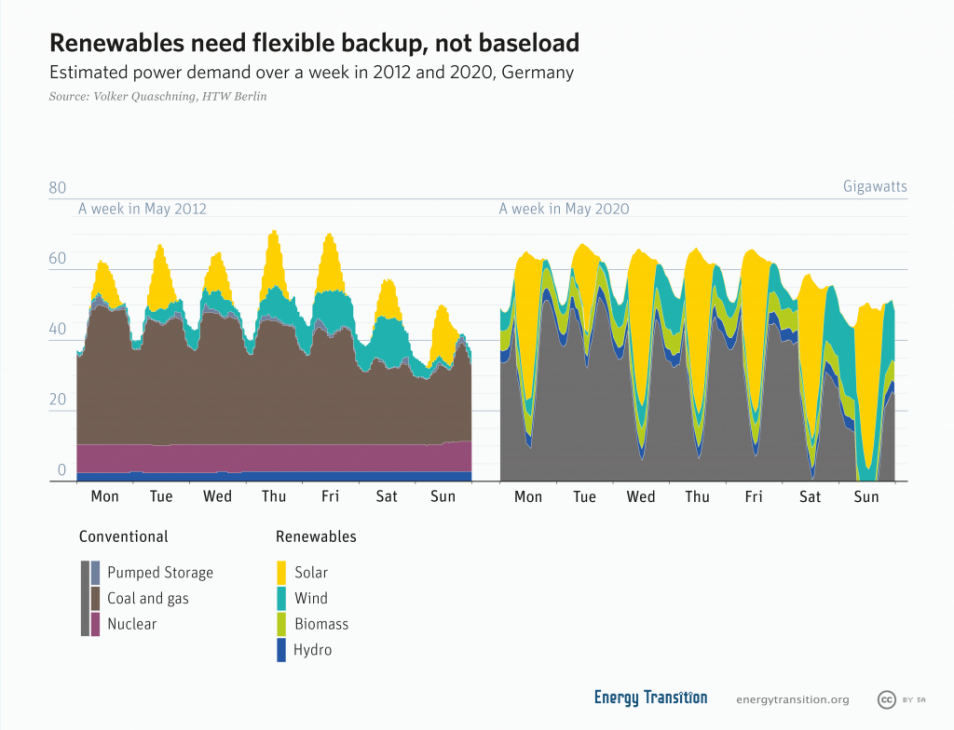


Figure 2.2: Renewable electricity is intermittent and hard to predict precisely. As the share of renewable electricity increases, the electrical grid needs to respond flexibly to meet demand.

storage.

rephrase, cite

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

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 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 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. 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.

cite

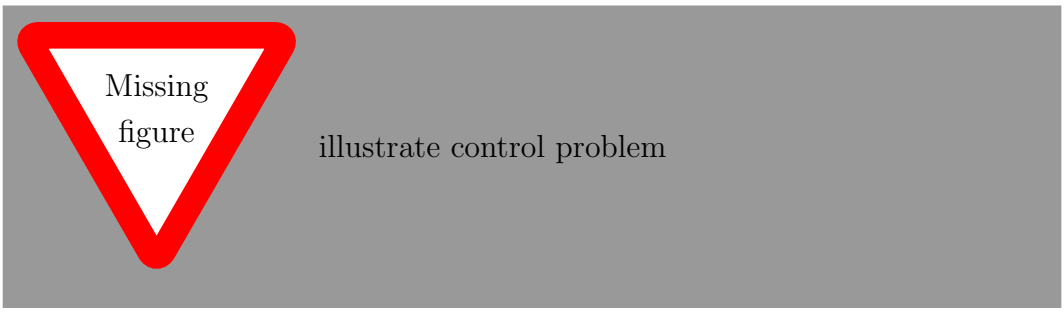
As stated before, the process of heating is being electrified. While this increases the load on the electric grid, it is also a chance to provide flexibility: When equipped with an intelligent and connected control system, the heating system can flexibly react to changes in price, using hot water tanks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

somehow mention coordination goals

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. When system dynamics are known and prices change predictably, for example with fixed rates per time of day, an optimal rule-based controller can be derived. A rule-based controller has the advantage of being transparent and reliable, but it's not flexible enough to react to changing system and pricing dynamics. Several adaptive control algorithms have been proposed for use in demand response.

cite and make sure this is true!



List some and cite

notes on this section: - focus more on buildings - use the following review paper: - Li et al. (2021), a review paper about energy flexibility in residential buildings

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 otherwise stated, this section is based on the textbook 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 can therefore also be called the 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. In complex problems, the desired behavior is not obvious. Therefore, designing the reward function is often non-trivial.

Maybe: Mention a few RL applications that can be used as examples

2.3.2 Q-Learning

Some environment states are preferable to others. Value Learning is a class of RL algorithms that builds on this intuition. From a trajectory of state-action-observation-reward tuples that were generated while following a policy π, a Value Learning algorithm assigns each state the expected future discounted reward that will be received when the agent, in state s, continues to act according to π:

$$v_\pi(s) = \mathbb{E}[G_t|S_t = s] = \mathbb{E}[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}|S_t = s], \text{for all } s \in S$$

Similarly, one can define the value of taking an action a in a given state s:

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

Mention exploration/expl dilemma here

This Action-Value Function or Q-function induces a greedy policy by evaluating $q(s,a)$ for the current state and all possible actions, and selecting the highest-expected reward action. For the optimal policy π_*, the greedy policy induced by q_{π_*} is π_* itself.

A process that iteratively improves an initial Q-function is Q-Learning. Its basic idea is that the Q-function can induce a better greedy policy than the one with which it was generated. A Q-Learning step consists of taking an action a in state s according to the current greedy policy and performing a Bellman update on $q(s,a)$, using $\max_a q(s',a)$. However, a slight modification needs to be made: To make sure all possible trajectories are explored, the ϵ-greedy policy is used instead, which acts randomly with a small probability of ϵ, and acts greedily otherwise.

Q = Quality of an action Focus less on the idea of value learning. Instead, explain Tabular Q-Learning better

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 standard online 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

$$[R_j + \gamma \max_a \hat{q}(s', a) - q(s, a)]^2$$

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.

$$Z_\pi(s, a) = p(G_t | S_t = s, A_t = a) = p\left(\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a\right) \quad (2.2)$$

$Z_\pi(s, a)$ 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.

Cruz et al. (2018) compare different action selection strategies, some of which dynamically adapt. cite one of them

2.4 Reinforcement Learning for Building Demand Response

- Motivation: RL is a class of control algorithms that has been successfully applied to DR. - Why is building DR a particularly good fit for RL? - Mention prior work: When has RL been applied to building demand response? - paper 1: - paper 2: - paper 3:
- (Mention related work: applied algorithms and their results) - (Other Frameworks and available data, including CityLearn, real-life applications)

2.4.1 CityLearn

1. Introduce and Explain CityLearn (cite) - What is it? - A simulation framework for constructing demand response environments for RL - What are its main contributions? - Provide an accessible entry point for RL researchers for DR (cite 2021 challenge paper) - Validate the usefulness for RL methods - Can use Real-World Data - Limited application as Benchmark suite 2. CityLearn has previously been approached in a number of ways. - paper 1 - paper 2 - paper 3

One particularly interesting framework for RL in demand response is CityLearn.

- !!! where does uncertainty come from in CityLearn !!!, - forecasting problems: - uncertainty about future occupant behavior (electricity demand) - uncertainty about future solar power production (weather forecasts) - uncertainty about future costs (price forecasts) - measurement uncertainty: - observations might be imprecise - coordination uncertainty: - uncertainty about other actors' strategy and current actions - reward uncertainty: - uncertainty about the consequences of actions - the observed reward might be imprecise - the observed reward might not be the intended reward ??? Which uncertainties need to be specially treated?

Motivation: What advantages would an uncertainty-aware strategy have? - better risk management (possibly) - better performance (possibly) - better interpretability and robustness (possibly)

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 by 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. 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.

cite

The simulation framework used by both the challenge and this thesis is CityLearn 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 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). These

are coupled with weather variables (Outdoor Temperature, Relative Humidity, Diffuse and Direct Solar Radiation). Future weather data is also provided as part of the data, with an offset of 6, 12 and 24 hours into the future. 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.

- Description: How does the data look? Show graphs, or reference graphs in appendix.
- Maybe: show a sample week of building data - Maybe: show a sample week of weather and CO2 data - Maybe: show seasonal variations

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[1] 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[2] 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 used the ML-Cloud 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.

how do I describe this?

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

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

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).

maybe: more details: - which networks? - which further tricks? (Target nets, Experience replay) - how are the uncertainties estimated?

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 wasted. 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 always stores 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. If battery charging starts too late, there is not enough time left to fully charge the battery. Therefore, the battery is charged as late as possible. 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.

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.

Algorithm 1 The Rule-Based Controller's Policy always stores observed excess solar power. Additionally, it ensures to charge the battery in the afternoon. After that, it tries to satisfy demand from the battery.

$a \leftarrow (\text{solar} - \text{load})/6.4$ ▷ Difference scaled to units of battery capacity
if $11 \leq \text{hour} \leq 15$ **then**
  $a \leftarrow \max(0.24, a)$ ▷ Slowly charge battery
end if
Ensure: $-1 \leq a \leq +1$
return a

3.3 Experiment

TODO: - Explain in more detail: What am I measuring, how will I evaluate the results? - Implementation details: runscript, slurm scheduling, etc.

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 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.

find the citation that says tuning epsilon is

Table 3.2: Hyperparameters, their tuning ranges or untuned values.

Hyperparameter	Value
Learning Rate	Tuned: $(0.1, 0.07, 0.03, 0.01, \ldots, 0.00001)$
Batch Size	Tuned: $(1, 2, 4, 8, \ldots, 256)$
Adam's ϵ	Tuned: $(1 \times 10^{-1}, 1 \times 10^{-2}, \ldots, 1 \times 10^{-9})$
Target Network Update Frequency	Tuned: $(4, 8, 16, 32)$
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

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.

3.3.3 Comparison of Tuned Algorithms

For each algorithm, I select the best combination of hyperparameters during tuning.

Following the analysis of Clements et al. (2020), I measure the fraction of non-greedy actions for each algorithm across a training run. This will illustrate 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. I also measure the time per training episode.

Chapter 4

Results

Incorporate Nicole's feedback! (see slack DMs)

4.1 Rule-Based Controller

Maybe: - How does it perform on the continuous action space? - On how many days does it needlessly fill the battery?

maybe: report this stat: How much solar electricity is used in both cases, how much electricity is purchased in both cases?

4.2 Discretization

In the pre-experiment on the effect of discretization, 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.

4.3 Hyperparameter Tuning

All 200 runs of ϵ-greedy DQN completed successfully. 10/100 runs of UA-DQN, all 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

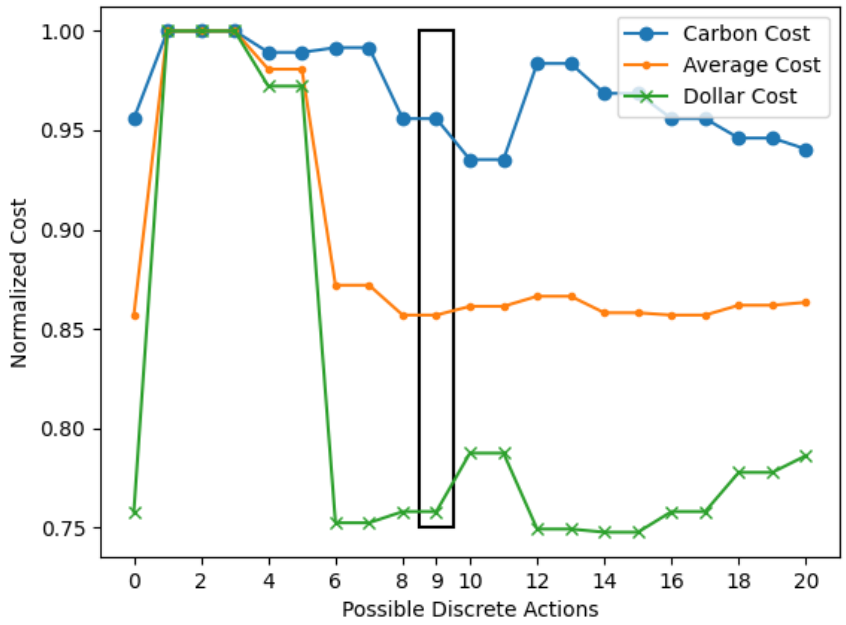


Figure 4.1: This graph shows the effect of the discretization resolution on the rule-based control algorithm. The coarsest resolution that does not significantly impact performance and includes the zero action is 9. In this graph, 0 possible actions means no discretization is applied.

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 UADQN 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, which corresponds to strategies that are close to the idle policy.

4.4 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.

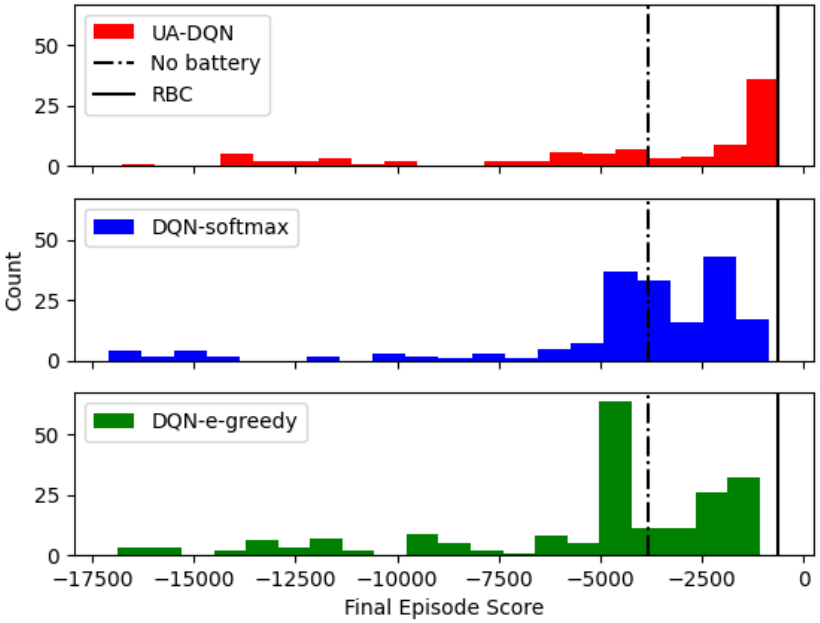


Figure 4.2: This histogram shows the final episode reward of all successful tuning runs. Every run represents one choice for the hyperparameters. The performance of the rule-based controller and the idle policy are shown for reference.

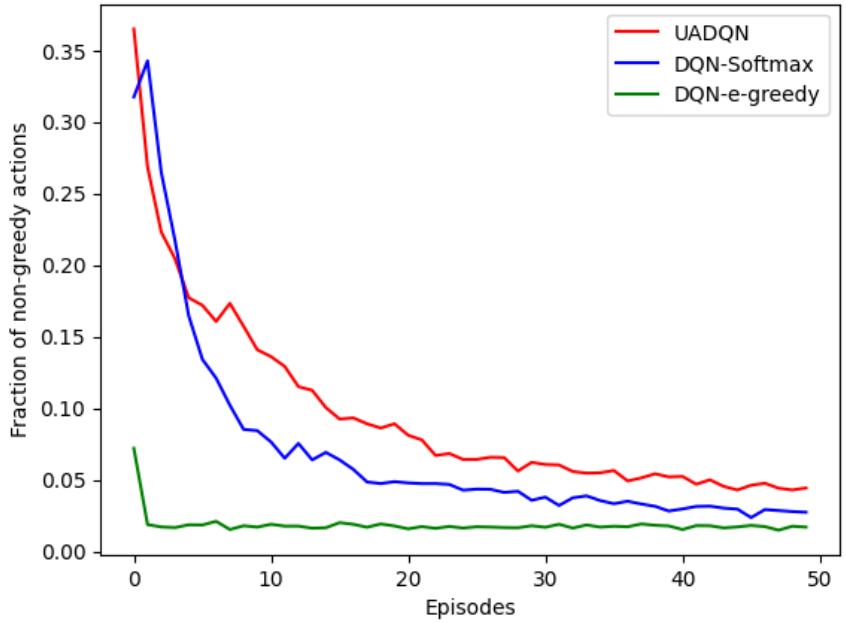


Figure 4.3: This graphic shows the exploration rate of tuned algorithms throughout the training process. It highlights the different action selection strategies employed by the different algorithms. UA-DQN keeps exploring for longer than the other strategies.

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
	Target Network Update Frequency	4	-0.11
DQN-Softmax	Learning Rate	3e-4	**-0.36**
	Batch Size	128	-0.18
	Adam's ϵ	1e-05	-0.02
	Target Network Update Frequency	4	0.16
ϵ-greedy DQN	Learning Rate	7e-05	**-0.33**
	Batch Size	4	-0.19
	Adam's ϵ	1e-08	0.10
	Target Network Update Frequency	16	0.14

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

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 metrics.

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

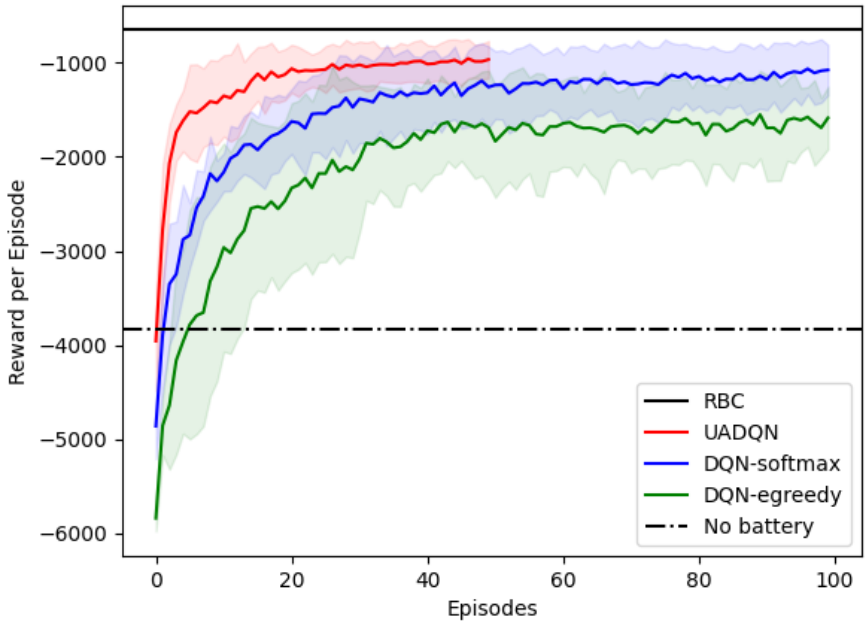


Figure 4.4: This graphic shows the performance during training of the three tuned algorithms. The shaded area shows the standard deviation over 10 runs with different random seeds. Tuned UA-DQN converges after fewer episodes than either DQN variant. UA-DQN was only trained for 50 episodes due to the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

Table 4.3: This table shows the mean performance of tuned algorithms when evaluated using their respective action selection policy for one episode on Building 1.

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 evaluated 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, tuned 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, instead simply assuming it does not change from the hour before. 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 vari-

maybe: report this stat: How much solar electricity is used in both cases, how much electricity is pur-

ants 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 by the fact that its minimum charging action of 0.24 in the hours before the high-price period starts is discretized into more or less useful values that for example don't fully charge the battery.

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 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. Adam's ϵ did not have a large effect on UA-DQN and DQN-Softmax, but tuning mattered on ϵ-greedy DQN.

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 all other runs seemed to converge. The DQN-Softmax algorithm failed to complete some runs with a batch size of 8. Even the tuned variant only completed 9/10 runs. This suggests some instability of the algorithm.

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. UA-DQN explores more throughout the whole 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 ex-

ploratory 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 helped UA-DQN, which was tuned for half as many runs as the other algorithms, so there was less opportunity for outliers to outperform more robust hyperparameters. 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. It depends on long-term, aggregate data and patterns on the scale of multiple hours. In contrast, a part of the rule-based policy's idea is to store precisely the amount of excess electricity generated, which is nearly impossible to do in CityLearn, but easy to implement in a real-world building energy system. This limits the effectiveness of a good long-term plan.

The environment used in the experiments only provided 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 buy and 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.

Maybe: Things I have not yet mentioned: - UA-DQN needs a discrete action space. - Talk about need for data: Interaction between replay buffer size and batch size. - What would a good setting for the risk-aversion parameter even be?

Chapter 6

Conclusion and Outlook

- 1 page - summarize again what your paper did, but now emphasize more the results, and comparisons - write conclusions that can be drawn from the results found and the discussion presented in the paper - future work (be very brief, explain what, but not much how)
Summary: - I evaluate UA-DQN on a custom task using CityLearn.
- Tuned UA-DQN requires fewer environment interactions than variants of DQN to learn a good policy. This suggests that efficient exploration is beneficial, and exploration in general is useful for Q-Learning on CityLearn. - Tuned UA-DQN does not outperform a simple rule-based policy.
Conclusions: - UA-DQN is a promising algorithm for environments where exploration is costly.
Future Work: - CityLearn with integrated control - Coordination task - Uncertain Weather forecasts. - how does UA-DQN with a nonstationary environment? How much does a learned policy transfer to a different building, for example?

In this thesis, I apply the UA-DQN algorithm to a custom building energy management task implemented in CityLearn. I find that tuned UA-DQN outperforms DQN in order to learn a good policy. This means things are harder to get right.

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, September 2020. 3.2.1, 3.2.2, 3.2.3, 3.3.3

Francisco Cruz, Peter Wüppen, Alvin Fazrie, Cornelius Weber, and Stefan Wermter. Action Selection Methods in a Robotic Reinforcement Learning Scenario. In *2018 IEEE Latin American Conference on Computational Intelligence (LA-CCI)*, pages 1–6, November 2018. doi: 10.1109/LA-CCI.2018.8625243. 2.3.4

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

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

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. 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

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

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

Selbständigkeitserklärung

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