Data Center energy agent RL optimization

Athanasios Katranis (2803183), Konstantinos Pasatas (2803568), Mert Unveren (2709757)

Vrije Univerisiteit Amsterdam, De Boelelaan 1105, 1081 HV Amsterdam
`https://vu.nl/nl`

Abstract. This paper establishes a mixed model approach for solving an energy optimisation approach with the goal of comparison. It sets a base heuristic to match and improve using tabular Q-learning where the state space is discretized. Following, a DQN is used to more granularly estimate the Q values and find hidden patterns. These results are compared using a validation to conclude on model performances. The results of the report show that tabular Q-learning performs best with the given setup. While the DQN is limited in the current setup, it provides promising results and warrants further investigation.

Keywords: Reinforcement Learning · Tabular Q-learning · DQN

1 Introduction

Driven by both financial concerns and environmental obligations, data centers are leading the way in putting new ideas into practice to efficiently handle their electrical needs. This report presents a reinforcement learning approach tailored for a data center with a daily energy requirement of 120 MWh, where the facility can flexibly buy or sell up to 10 MWh of electricity per hour. The system also includes a storage mechanism capable of carrying over a maximum of 50 MWh between days, enabling strategic energy decisions that reduce operational costs and enhance overall energy stability. By dynamically adjusting purchasing and selling strategies, the proposed method aims to optimize financial outcomes for modern data center operations.

We implement a tabular Q-learning solution where the state space comprises storage level, electricity price, hour of day, and current day. The action space represents (continuous) buying/selling decisions between -1 and 1. The environment enforces specific mechanisms for forced buying when daily requirements aren't met. The reward structure is based on market transactions where selling actions receive 80% of the current market price.

2 Exploratory Data Analysis

To inform the discretization strategy and reward shaping mechanisms, we conducted exploratory data analysis on electricity price trends.

The hourly price distribution (Fig. 1) along with the average hourly price revealed (Fig. 2) consistent daily cycles, with prices reaching a peak around midday (Hour 12) and hitting a low during the early morning hours (Hour 5).

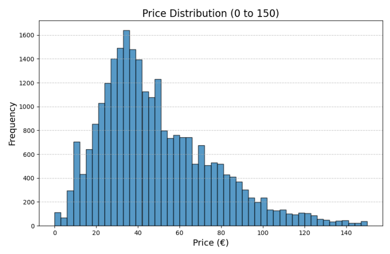

Fig. 1. Price distribution histogram, indicating frequent low prices and rare high spikes.

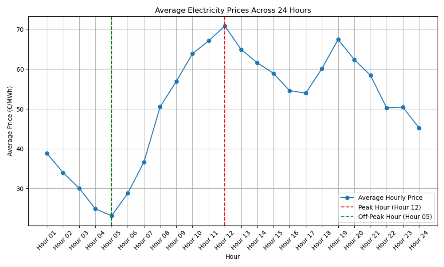

Fig. 2. Average electricity prices across 24 hours, showing peak and off-peak periods.

The Bollinger bands and correlation heatmap (Figure 3) further highlighted the short-term dependencies in price movements, showing that electricity prices were strongly correlated within small time windows. This indicated that a simple threshold-based approach would be insufficient for effective trading (Threshold-based agent). Instead, it motivates the use of a moving average-based strategy in reward shaping, allowing the agent to make dynamic decisions based on recent trends.

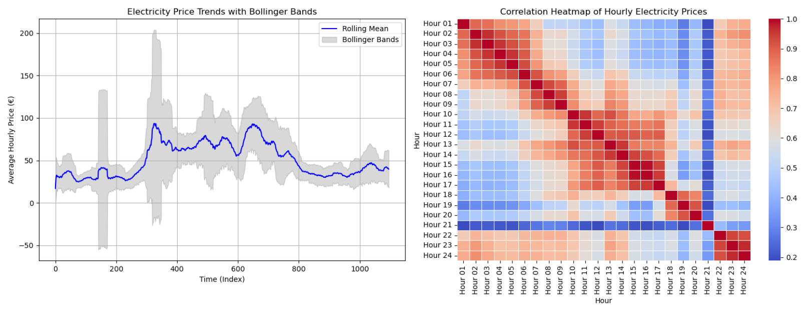

Fig. 3. Electricity price trends with Bollinger bands and correlation heatmap.

The seasonal and monthly boxplots (Figure 4) revealed that winter and autumn exhibited higher electricity prices on average compared to spring and summer. This suggests that electricity demand and pricing patterns are influenced by seasonal effects. Given this observation, an alternative approach could have been discretizing the day into seasonal categories, allowing the agent to incorporate seasonal trends into its decision-making.

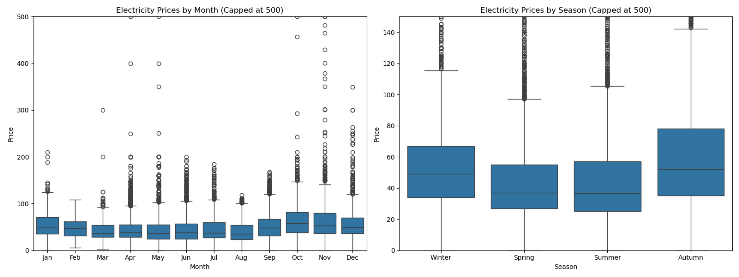

Fig. 4. Boxplots of electricity prices by month and season, highlighting seasonal irregularities.

The maximum and minimum hourly price analysis (Figure 5) showed extreme price spikes at certain times, highlighting the need for strategic energy storage. This highlights the need for smart storage management. Gradually storing energy instead of last-minute buys at peak prices would help in optimizing trading and reduce costs.

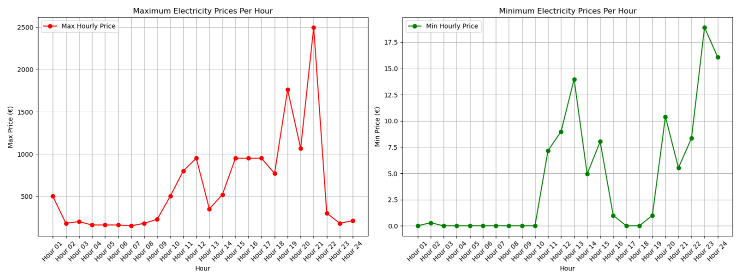

Fig. 5. Maximum and minimum electricity prices per hour, showing price volatility.

3 Methodology and implementation

3.1 Agent Types

Three distinct types of agents are used to tackle this problem: **(1)** A simple threshold-based agent uses hard-coded rules for buying or selling if certain price or storage thresholds are reached (sell if price > 140 and buy if price < 40); **(2)** A tabular Q-learning agent updates a discrete Q-table indexed by state-action pairs; and **(3)** A Deep Q-Network (DQN) agent leverages a neural network to estimate Q values. In the code shown, `DQNAgent` exemplifies the neural approach by storing experiences in a replay buffer, applying gradient-based updates, and maintaining a separate target network for stable learning.

3.2 Exploration-Exploitation Strategy

All agents balance exploration and exploitation through an ϵ-greedy strategy. After a lot of experimentation we concluded that we will initialize ϵ to 0.3 and a decay factor of 0.99 after each episode, and it is bounded below by a minimum value of 0.01. Thus, early in training, the agent chooses random actions with probability $\epsilon = 0.3$, encouraging exploration of novel behaviors. Over time, ϵ shrinks to 0.01, allowing the agent to take advantage of its learned policy more frequently. We made this choice because we realized that even when we had a $\epsilon = 1$ it will eventually lead to the same policy, but with a lot of extra training.

3.3 Basic Threshold Agent

The basic treshold agent buys when the price is below 40 and sells when the price is above 140, it reaches a total reward of -4,035,829.50 on the validation set and has a total of 6,579 of forced actions, with 970 mwh as storage waste. This means that in general the agent has quite a bit of forced actions although as a percentage of the total buys of 37%, meaning about a third of the actions were forced. In general this will be our baseline we will compare against.

3.4 Tabular Q-learning

Structure The Tabular Q-Learning approach maintains a Q-table to approximate the action-value function $Q(s, a)$, where each discrete state-action pair is assigned a value representing the expected future reward. The agent learns by iteratively updating these values through the Bellman equation, improving its decision-making over time.

The Q-table in our implementation is a 4D array of shape (storage bins, price bins, hours, actions), where each dimension corresponds to a discretized state variable. The agent selects actions using an epsilon-greedy policy, balancing exploration and exploitation. The learning rate ($\alpha = 0.1$) determines how much new information updates the Q-values, while the discount factor ($\gamma = 0.99$) controls the importance of future rewards. Exploration begins with $\epsilon = 0.3$ and gradually decays ($\epsilon_{\text{decay}} = 0.995$) until reaching a minimum threshold of $\epsilon_{\text{min}} = 0.01$.

During each step, the agent observes the state (s), selects an action (a), and receives a reward (r) based on the environment's response. The next state (s') is then observed, and the *Q-value update* is performed as:

$$Q(s, a) \leftarrow Q(s, a) + \alpha \left(r + \gamma \max_{a'} Q(s', a') - Q(s, a) \right)$$

Discretization Tabular Q-learning requires discretizing continuous state variables to create a manageable state space. The final discretization resulted in 3,456 states, balancing granularity and computational efficiency. Storage levels (0–170 MWh) were divided into 12 bins, ensuring the agent could differentiate between low, medium, and high reserves without excessive complexity. Electricity prices (0–2500 €/MWh) were discretized into 12 bins with non-uniform spacing, using finer bins at lower values (e.g., 0-10, 10-20 €/MWh) and wider bins at higher values (e.g., 100-250, 500-1000 €/MWh). The reason behind this way of discretization lies to the distribution of the price, where lower prices were more frequent and fluctuated in smaller increments, while higher prices were rarer but varied significantly. The hour of the day was kept continuous (1-24) without further discretization, as time plays an important role in energy trading. Seasons were initially included in the state, computed by using the day in the dataset, but caused issues in spring and summer 2007, where the agent sold low and bought high. This suggested overfitting to seasonal trends rather than learning a stable strategy. By removing seasons from the state space, we improved the agent's stability and prevented it from making irregular trading decisions based on seasonal anomalies. Finally, we chose three distinct options for the action space: purchase (1), hold (0), and complete sell (-1). This made decision-making straightforward and guaranteed the agent could handle trades efficiently.

Reward Shaping Our reward shaping process was designed iteratively, starting from a simple framework and progressively refining the rewards/penalties based on observed agent behavior. Initially, we replaced the threshold from the simple threshold-base agent with moving average-based decision-making. This allowed the agent to learn a more dynamic strategy to trade. We rewarded sales when prices were significantly high and purchases when the current price was much lower than the historical trend. The size of the chosen window was 8 hours based on trial and error. This led to a notable improvement in performance, as the agent started to buy low and sell high on average.

However, after analyzing the action plot based on the policy, we noticed a large forced buy period, where the agent failed to purchase energy strategically and was forced to buy in order to meet the daily demand. To address this, we introduced a forced buy penalty, in order to discourage the agent from entering stages where it would have to make suboptimal purchases due to lack of daily resources. The forced buy penalty checks if the agent can still meet the 120 MWh daily demand withing the remaining hours at the maximum rate of 10 MWh per hour. If not, a penalty of -50 is applied. Additionally, another issue with the moving average approach was that the agent sold too frequently, which also led to the forced buy problem. To address this, we introduced a sell restriction penalty (-20), preventing sales that would make it impossible to meet the 120 MWh target within the remaining hours. This way, the agent learned to hold its position when necessary.

Furthermore, in order to ensure the agent gradually accumulates energy instead of making last-minute bulk purchases, we introduced storage rewards at key points throughout the day. The agent receives a +20 reward at 12:00 PM if it has stored at least 70 MWh, a +30 reward at 8:00 PM for reaching 100 MWh, and a +50 reward at 1:00 AM if it meets the 120 MWh daily target. By rewarding intermediate storage milestones, the agent learns to manage its energy strategically, leading to a more stable and cost-effective policy.

Finally, we observed that force buys were not always harmful, as electricity prices sometimes dropped later in the day. Holding energy for too long could result in missed chances to sell at high prices earlier and buy back at a lower cost later. To encourage the agent to take advantage of high prices earlier in the day, we added an early sell bonus. If the agent sells before hour 15 and the price is at a local peak (at least 20% higher than the last two-hour average), it earns a small reward (+5.0). This helps the agent sell at optimal times, preventing unnecessary hoarding while still allowing it to benefit from potentially lower prices later in the day if a forced buy happens.

3.5 Deep Q-Network

Structure The Deep Q-Network (DQN) approach uses a neural network to approximate the action value function $Q(s, a)$ for each discrete action bin -1, 0, and 1. The key hyperparameters that we used are a learning rate ($\alpha = 0.001$), an initial discount factor ($\gamma = 0.0$) that progressively increases to 0.998 after a power curve, an initial exploration rate ($\epsilon = 0.3$) with a decay rate of 0.99, and a minimum exploration threshold ($\epsilon_{\min} = 0.01$). The network architecture consists of a fully-connected multilayer perceptron with two hidden layers, each containing 128 neurons and utilizing ReLU activations, culminating in an output layer matching the number of action bins. To enhance training stability, the agent maintains two separate networks: a primary model for generating current Q-value estimates and a target model for computing stable target Q-values during updates.

Also, We are using a Prioritized Experience Replay (PER) mechanism, utilizing a SumTree structure with a maximum capacity of 10,000 experiences. This buffer prioritizes experiences based on their temporal-difference (TD) error, controlled by the hyperparameters $\alpha = 0.6$ and $\beta = 0.4$, with β steadily increasing to 1.0 to fully compensate for the non-uniform sampling bias. The Mini-batches have a size 64 and they are sampled from the replay buffer, incorporating n-step returns with $n = 3$. Each sampled batch undergoes the computation of temporal difference loss using a mean squared error (MSE) criterion. The Adam optimizer is then employed to backpropagate and update the network parameters, minimizing the weighted loss.Lastly, regarding the target network we are using a tau ($\tau = 0.1$) parameter, applied every 100 training steps to ensure gradual and stable convergence.

Reward Shaping The DQN agent employs a structured reward shaping strategy to optimize energy trading while ensuring safe storage levels. While it follows similar core principles to Tabular Q-learning, it introduces refinements that leverage short-term and long-term price trends for more adaptive decision-making.

To maintain storage safety, the agent is penalized for selling below 10 MWh or buying above 190 MWh, with penalties scaling based on price to discourage inefficient trades. End-of-day storage optimization rewards maintaining storage between 80-150 MWh (+40) while imposing -40 penalties for excess (>170 MWh) or shortages (<50 MWh), ensuring energy stability for the next day.

For long-term (24h) trading, rewards scale with price differences and storage levels, with multipliers reaching 18x for large reserves. Emergency buying (<80 MWh) allows above-average price purchases to prevent critical shortages. Short-term (4h) trading encourages opportunistic buying/selling with multipliers up to 4.5x, while emergency short-term purchases (<60 MWh) trigger urgency-based premiums to prevent sudden depletion.

A priority hierarchy ensures that storage safety and end-of-day balance take precedence over trading opportunities, followed by long-term and short-term price trading, with emergency interventions activated only when necessary. This structured reward design allows the DQN agent to dynamically adapt to price trends, maintaining stable reserves while maximizing trading efficiency.

4 Results and analytics

4.1 Tabular Q learning

To assess the generalization of the Tabular Q-learning agent, we evaluated its performance on a 720-day validation set using different trained Q-tables. The results show a gradual improvement in total reward as training progressed, with the lowest total reward of -4,256,367.62 occurring at 1,400 episodes, while the best-performing model at 3,200 episodes achieved a total reward of -3,691,216.88.

The trend indicates that the agent learns a more effective policy over time, reducing losses and optimizing its trading behavior. However, performance fluctuates after 3,200 episodes, suggesting potential overfitting or instability in later training stages. These results confirm that Q-learning successfully adapts to unseen data, though careful tuning of training duration is necessary to prevent performance degradation.

Episodes	**Total Reward**
3200	-3,691,216.88
3300	-4,118,929.44
3400	-3,890,197.58
3500	-3,986,215.18
3600	-4,069,709.04
3700	-3,871,232.50
3800	-3,808,503.20
3900	-3,884,404.46
4000	-4,117,834.54

Table 1. Total reward on the validation set for different training durations.

The tabular Q-learning plot 6 clearly depicts a clear upward trend, indicating that the agent is progressively learning an improved policy over time. The cumulative average reward, represented by the red dashed line, steadily increases, suggesting effective optimization of the reward function. While the general trend signifies policy improvement, the continued fluctuations indicate that some level of exploration can be still achieved. Additionally, techniques such as reward normalization, experience replay, or batch updates could mitigate fluctuations and accelerate convergence. Overall, the results indicate that the Q-learning agent is effectively learning, though further refinements may be required to reduce variance and ensure a more stable policy.

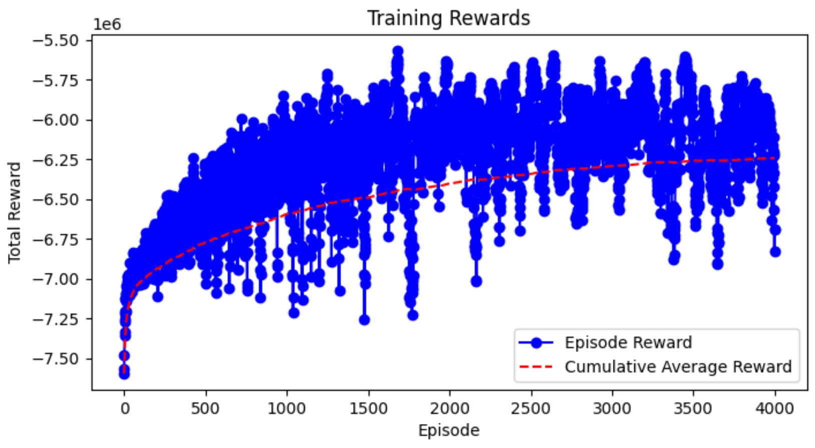

Fig. 6. The plot displays the reward per episode and the cumulative reward.

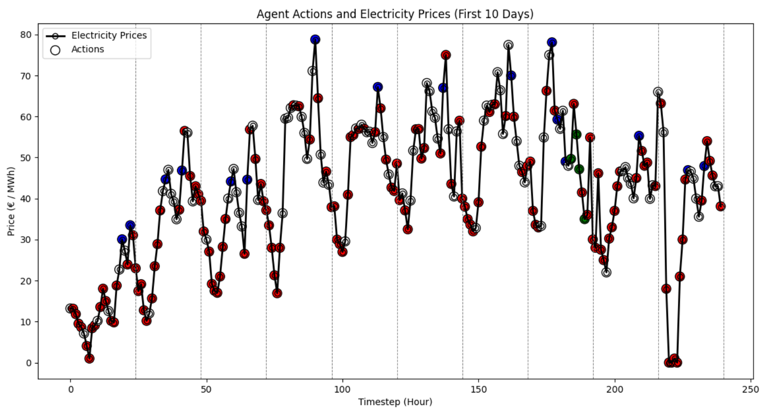

Fig. 7. Agent actions (buy, sell, hold) overlaid on electricity price fluctuations for the first 10 days. Green dots indicate buy actions, blue dots indicate sell actions, and red dots represent hold decisions.

The action plot in Figure 7 illustrates the trading behavior of the Tabular Q-learning agent over the first 10 days, with electricity prices plotted as a black line and the agent's buy (red), sell (blue), hold (white), and force buy (green) actions.

The agent buys (red) when prices are low and sells (blue) when prices are high, showing that it has learned to trade strategically. Hold actions (white) appear when prices are stable, suggesting the agent avoids unnecessary trades, reducing the risk of inefficient decisions. Additionally, on day 8, the agent made a forced buy. Since earlier prices were high compared to later hours, the agent's decision was reasonable. Another trend in the plot is that the agent buys mostly in the early hours, which aligns well with the reward for maintaining early storage levels. This confirms the agent's strategy of storing energy gradually instead of buying last minute.

Overall, the plot highlights how reward shaping guided the agent's behavior, with most decisions aligning logically with the designed reward function.

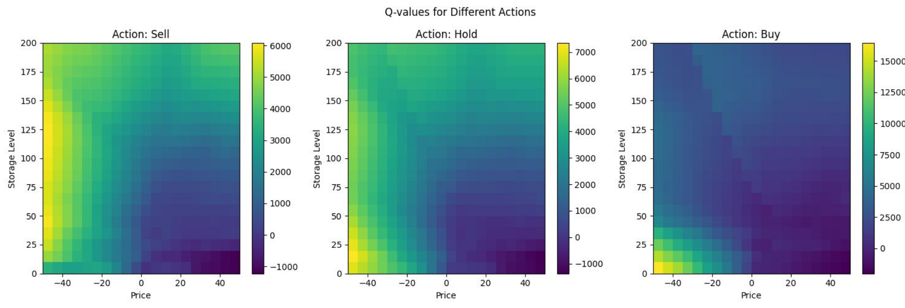

Fig. 8. Q-values for different actions across storage levels and price variations.

4.2 DQN

The Q-value heatmaps in Figure 8 depicts the action-value function for Sell, Hold, and Buy decisions based on storage levels and price.The Sell action left assigns the highest Q-values to high storage levels and very negative price values, indicating that selling is optimal when storage is high, even at lower prices. The Hold action the middle subplot remains relatively balanced, with higher Q-values in regions where price fluctuations are moderate.Lastly, the Buy action right subplot has the highest Q-values when storage is low, and prices are very negative, showing that the agent has learned to buy when prices are at their lowest.

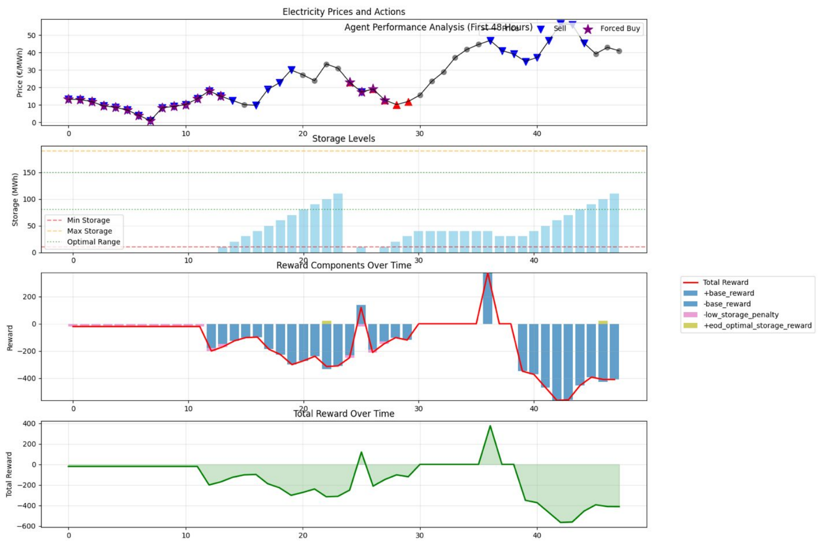

Fig. 9. Agent Performance Analysis

The agent performance analysis in Figure 9 provides a detailed breakdown of its decision-making process over the first 48 hours. The top plot illustrates the price and the corresponding agent actions. The agent generally buys when prices are low and sells when prices rise, which is a reasonable trading strategy. The second plot shows the storage level over time, showing an increase in storage, particularly when the prices are low. The third plot breaks down reward components, with the total reward fluctuating due to penalties and bonuses related to storage constraints. The final plot depicts the cumulative reward, highlighting periods of profitability but also frequent negative dips, suggesting that the agent still struggles with optimal decision-making. Overall, the agent demonstrates a good strategic trading behavior, further refinements in reward shaping could enhance his performance.

Table 2. Validation Results and Reward Component Breakdown

Metric / Component	**Average per Step**	**Total Value**
Total Base Reward	–	-3,993,758.24
Average Reward per Step	–	-228.27
Number of Steps	–	17,496
base_reward	-228.27	-3,993,758.24
eod_optimal_storage_reward	0.83	14,580.00
low_storage_penalty	-3.26	-57,080.00

5 Conclusion and Discussion

In this study, we explored different reinforcement learning approaches for electricity trading, including a simple threshold-based agent, Tabular Q-learning, and a Deep Q-Network (DQN). The threshold-based agent provided a baseline but struggled with forced actions and inefficient storage management. The Tabular Q-learning agent showed clear improvements over time, learning to optimize trading decisions while balancing storage constraints. However, performance fluctuations suggested that further refinements in discretization and reward shaping could enhance stability. The DQN agent leveraged deep learning techniques and prioritized experience replay to generalize across complex state spaces although it did not achieved the best result. The Q-value heatmaps and performance analysis indicated that the agent learned rational trading behaviors, such as buying at low prices and selling at high prices, but still faced challenges in fully optimizing storage levels and avoiding penalties.

The DQN can be further optimized by hyperparamater tuning, better reward shaping, deeper NN, different activation functions, possibly better gradient search for this specific problem. This part leaves much to explore and improve upon. Overall, our findings demonstrate that reinforcement learning is a promising approach for automated electricity trading from a data center, with structured reward shaping playing a critical role in guiding optimal agent behavior.