Project Reinforcement Learning

Group 11
Alexia Spinei, Claudia Domènech Farré, Satiga Godrie

January 30, 2026

1 Introduction

For this project, we studied the operational control of a pumped-storage hydroelectric dam under uncertain electricity prices. This problem is inherently sequential, as current decisions affect future storage availability and profit opportunities, making it well-suited to a reinforcement learning formulation. At each hour, an operator decides whether to store energy by pumping water, generate electricity by releasing water, or remain idle. The objective is to maximize cumulative profit over a given time series of electricity prices. We first establish a simple rule-based baseline and then apply reinforcement learning methods to learn improved control policies.

2 Methods

2.1 Environment

We model the dam operation as a discrete-time Markov Decision Process (MDP) with an hourly time step, implemented in a Gymnasium environment. At each time step, the agent observes the current system state and selects an action that determines the dam's operation during the next hour.

The observation consists of the discretized reservoir level, the current electricity price (optionally normalized using a rolling 168-hour percentile), and temporal features derived from the timestamp, including hour-of-day, weekend indicator, and season. The agent has no access to future prices.

The action space is discrete, $\mathcal{A} =$ idle, sell, buy, and actions are subject to physical flow and storage constraints. The maximum hourly water volume is $V_{\max} = 5\ \mathrm{m}^3/\mathrm{s} \times 3600 = 18{,}000\ \mathrm{m}^3$.

Energy Model and Reward

The potential energy associated with moving a water volume V is $E(V) = \frac{\rho g h V}{3.6\times 10^9}$ (MWh), where $\rho = 1000\ \mathrm{kg/m}^3$, $g = 9.81\ \mathrm{m/s}^2$, and $h = 30$ m. Electricity generation has efficiency $\eta_{\text{turbine}} = 0.9$, while pumping has efficiency $\eta_{\text{pump}} = 0.8$.

The reward corresponds to hourly profit or cost:

- Sell: $r_t = P_t \cdot (\eta_{\text{turbine}} \cdot E(V))$
- Buy: $r_t = -P_t \cdot (E(V)/\eta_{\text{pump}})$
- Idle: $r_t = 0$

where P_t denotes the electricity price at time t. Storage updates deterministically based on the selected action, and each episode terminates at the end of the available price time series.

2.2 Baseline Model

As a baseline, we implement a simple rule-based threshold policy that uses only the current electricity price to make decisions. This policy provides an interpretable benchmark for evaluating reinforcement learning methods.

Electricity prices are classified into low, medium, and high regimes using empirical quantiles of the price distribution. The buy and sell thresholds are defined as the 33rd and 67th percentiles, respectively, and are

computed once and fixed during evaluation to prevent future information leakage. At each hour, the policy buys when the price is below the lower threshold, sells when the price is above the upper threshold, and remains idle otherwise.

The baseline is evaluated by rolling it forward over the validation price series, recording cumulative profit, storage levels, and actions. Performance is assessed using cumulative profit and qualitative analysis of operational behavior through time-series visualizations.

2.3 Tabular Q-Learning

We use tabular Q-learning, a model-free algorithm that maintains a table $Q(s, a)$ storing the expected cumulative reward for taking action a in state s. Q-values are updated using the Bellman equation:

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

where α is the learning rate, γ is the discount factor, r is the immediate reward, and s' is the next state.

For Q-values to reliably estimate expected returns, each state-action pair must be visited multiple times; a state visited once reflects a single trajectory rather than an average over outcomes. This requirement motivates keeping the state space compact through careful feature engineering. With our 1,440-state representation, most mid-range states are visited sufficiently often during training.

We use ε-greedy exploration, where the agent selects a random action with probability ε and otherwise selects the highest-value action. We decay ε over episodes, shifting from exploration toward exploitation as Q-values stabilize.

Parameters α and γ are tunned in detail later on (explained in the Parameter Tuning section 2.7).

2.4 Dataset Analysis

Our dataset comprises 3 years of hourly electricity prices (2007–2009), totaling approximately 26,280 time steps. Prices range from near-zero to €2,500, though values above €200 occur in less than 1% of hours. The heavily right-skewed (see Figure 7 in Appendix) distribution motivates quantile-based discretization.

Figure 1a shows distinct daily patterns: Night hours (1–6) have the lowest mean prices, Midday (10–16) the highest, and Evening Peak (17–21) the highest mean and volatility. This pattern suggests buying during Night and selling during Midday or Evening Peak. Figure 1b confirms that Evening Peak hours have the highest coefficient of variation. These patterns informed our hour period groupings.

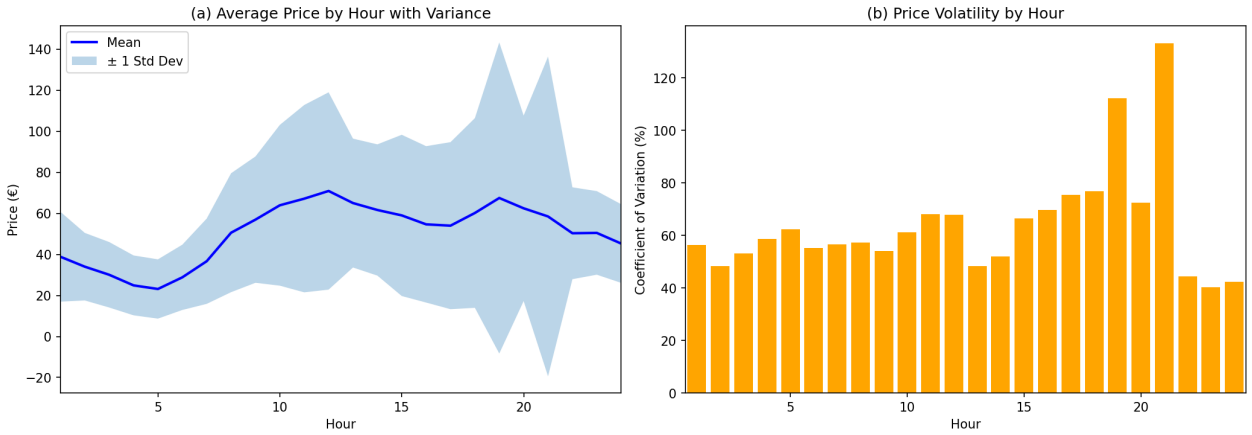


Figure 1: (a) Average price by hour with ±1 standard deviation. (b) Price volatility (coefficient of variation) by hour.

Prices also vary by season (Figure 2b): Fall and Winter have the highest average prices, while Summer is lowest. The interaction between season and hour period (Figure 3b) shows that optimal strategy may differ between, for example, Summer and Fall evenings, motivating inclusion of both features in the state

representation. Weekend prices are generally lower than weekday prices (Figure 2c); we capture this with a binary indicator rather than day-of-week, trading granularity for increased state visitation. Figure 3a confirms this distinction captures the primary day-level variation. Combining these patterns, fall and winter evenings on weekdays represent the strongest expected selling opportunities, while spring and summer mornings (particularly on weekends) offer favorable buying conditions.

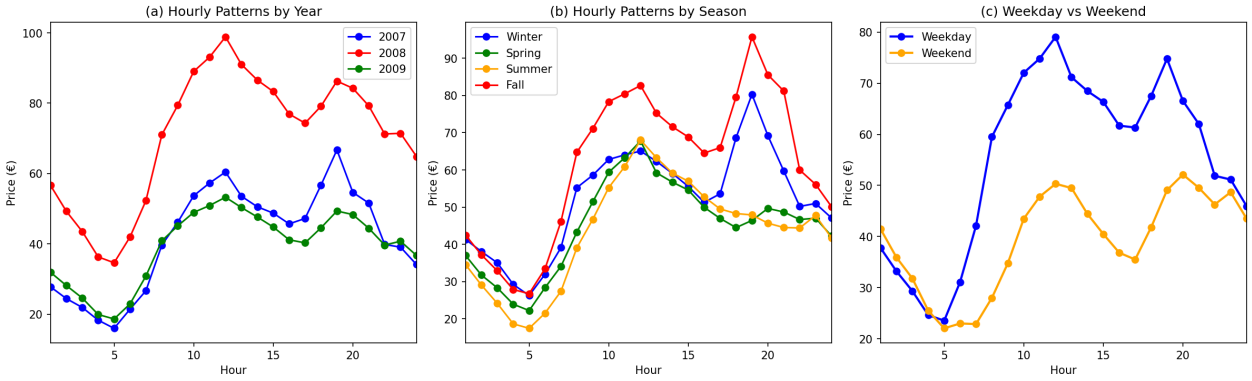


Figure 2: Hourly price patterns by (a) year, (b) season, and (c) weekday versus weekend.

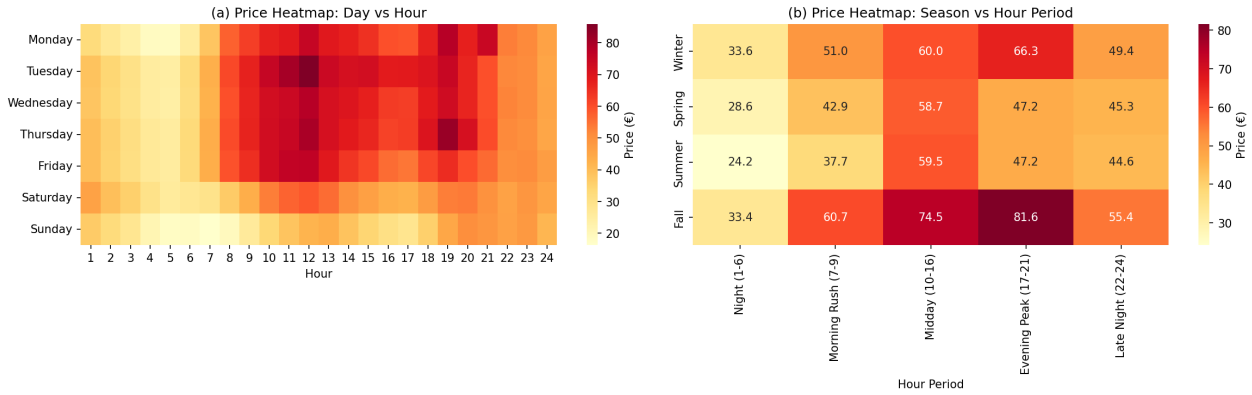


Figure 3: Heatmaps of mean prices by (a) day of week and hour, and (b) season and hour period.

The 2008 energy crisis caused a 69% increase in average prices compared to 2007 and 2009; Figure 2a shows this shift persists across all hours (see also Figure 8 in Appendix for seasonal breakdown). An agent trained on 2007 might treat €50 as high, while in 2008 this would be below average, motivating our use of adaptive price normalization.

2.5 Feature Engineering

For tabular Q-learning, each state is independent with no generalization between similar states. We represent the state as $s = (\text{storage\_bin}, \text{normalized\_price\_bin}, \text{hour\_period}, \text{is\_weekend}, \text{season})$, yielding $6 \times 6 \times 5 \times 2 \times 4 = 1{,}440$ state spaces.

Storage was discretized using 6 equal-width bins, while price was discretized into 6 quantile-based bins to ensure roughly equal observations per bin. To assess whether this was appropiate, we monitored state visitation counts during training and defined a state as well-visited if it was encountered at least 1000 times, corresponding to approximately 300 samples per action. This criterion ensured that most discretized states were visited often enough to learn reliable Q-values, without making the tabular state space overly sparse.

The 24 hours are grouped into 5 periods (Night 1-6, Morning Rush 7-9, Midday 10-16, Evening Peak 17-21, and Late Night 22-24) based on similar mean prices and volatility, reducing state space by a factor of

4.8 while preserving decision-relevant structure. The remaining two dimensions distinguish weekday versus weekend behavior and the four meteorological seasons.

To address inter-annual variability, we normalized prices using a rolling 168-hour percentile, producing values in $[0, 1]$ representing price rank relative to recent history. We used a 168-step rolling window because the data is hourly and 168 hours corresponds to one full week. Using a weekly window provides a stable local reference for "high" and "low" prices without mixing different regimes, which improves generalization over time.

2.6 Reward Shaping

The agent receives non-zero rewards only when buying or selling, creating a credit assignment challenge: buying has negative immediate reward but enables profitable selling later. In early experiments, we observed that the learned policy strongly preferred selling over buying, indicating that the immediate negative reward discouraged exploration of the buy action. Our initial approach penalized entering peak periods with low storage, but this did not improve performance as the agent optimized the shaped reward directly rather than the true objective.

We then used potential-based reward shaping, which preserves the optimal policy by adding the potential-reward to the next-state its returned reward. We define potential based on storage value: potential_reward = (storage_change) $\times$ price $\times$ scale. When buying, storage increases, yielding a positive shaping reward that partially offsets the immediate cost and vice versa reducing the profit for selling stored water. This flattens the reward landscape, reducing the extreme negative signal that would otherwise discourage the agent from exploring the buy action. To establish the optimal scale for the reward shaping, we conducted a scale-sweep with a fixed gamma (0.9) and learning-rate (0.1). Per scale-value 10 seeded agents (ranging from 1 to 10) were trained to reduce stochasticity and test for robustness. The agents with scale 50 performed best on the validation set, also scoring the lowest variance, indicative of generalizability.

Scale	Mean (€)	Std	% Baseline	Scale	Mean (€)	Std	% Baseline
0.5	29,648	2,530	74%	**50**	**52,590**	**1,646**	**131%**
10	34,062	2,712	85%	70	47,053	2,549	118%
30	44,023	3,233	110%	100	13,911	3,277	35%

Table 1: Scale sweep results ($\gamma = 0.9$, 160 episodes, 10 seeds). Scale 50 achieves 131% of baseline with lowest variance.

2.7 Parameter tuning

Phase 1: One-Dimensional Sweeps

We first sweep one parameter at a time to establish a stable baseline and see each parameter's isolated effect. We test $\alpha \in \{0.005, 0.01, 0.05, 0.1, 0.5\}$ and $\gamma \in \{0.90, 0.95, 0.99, 0.995, 0.999\}$ using seeds $\{42, 123, 456\}$, reporting mean validation PnL (Profit and Loss), standard deviation, and Coefficient of variation (CV). We see in figure 9 that the best are $\alpha = 0.01$ (with $\gamma = 0.99$ fixed) and Figure 10 $\gamma = 0.90$ (with $\alpha = 0.1$ fixed).

Phase 2: Two-Dimensional Grid Search

We then run a 2D grid to capture parameter interactions that 1D sweeps cannot reveal. The parameters that we chose here were more specific and closer to the ones that provided a good performance in the Sweep. The grid uses $\alpha \in \{0.05, 0.1, 0.2, 0.5\}$ and $\gamma \in \{0.93, 0.95, 0.97, 0.99, 0.995\}$ (20 configs), again with the same three seeds. The heatmaps (Appendix Figures 11 and 12) show the best region at $(\alpha = 0.05, \gamma = 0.97)$ with mean PnL $>$ €50,000 and low variance.

Phase 3: Adaptive Episode Allocation

Small learning rates converge more slowly, so we extend training to 300 episodes for $\alpha \in \{0.005, 0.01\}$ (others: 250) to keep comparisons fair by ensure reaching convergence as well. Validation PnL is evaluated every 20

episodes with a greedy policy ($\varepsilon = 0$) to avoid exploration noise.

2.8 Robustness Validation

To confirm results are not seed-specific, we re-train two candidates with 10 different seeds (ranging from 1 to 10): Model 1 ($\gamma = 0.90, \alpha = 0.01$) discovered by conducting the 1D sweeps and Model 2 ($\gamma = 0.93, \alpha = 0.05$) discovered by conducting the grid search. Both models were trained for enough episodes ensuring convergence, Model1 with 300 episodes and Model2 with 250 episodes, with the same 10 seeds to enable direct comparison and make sure it not just reflected 1 lucky training due to a favorable stochastic nature.

Figure 4a shows training rewards across seeds: both models exhibit negative training rewards due to exploratory actions, but Model 2 (orange) achieves less negative rewards, and this therefore indicates more efficient learning. The small confidence bands (± 1 std) demonstrate high reproducibility.

Figure 4b shows an important difference: Model 1 (blue) slowly converges to €48,000 by episode 250, while Model 2 (orange) quickly reaches €55,000 by episode 50 and maintains stable performance. Figure 5 also shows this advantage across all of the 10 seeds: Model 2 achieves €59,777 $\pm$ 2,192 (mean $\pm$ std), outperforming Model 1's €48,494 $\pm$ 459 by 23%. The relatively small error bars rule out the random chance.

3 Results

3.1 Final Model Selection

We select the final model in two steps to balance performance and efficiency.

Stage 1: The 2D grid identifies Model 2 ($\gamma = 0.93, \alpha = 0.05$) with mean PnL €50,224 and std €1,515 (Appendix Figures 11–12).

Stage 2: Figure 4b shows this model reaches its peak by episode 60, after which validation PnL plateaus with minimal improvement. Training beyond this point provides not many gains while increasing computational cost. So we set 60 episodes as the default parameter for thiss model.

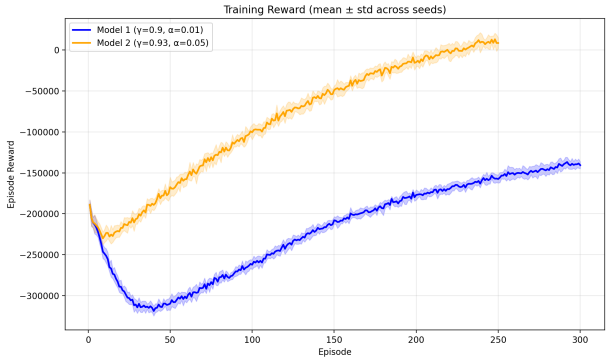
Final model: $\alpha = 0.05$, $\gamma = 0.93$, 60 episodes (CV < 0.04 across 10 seeds).

3.2 Performance Analysis

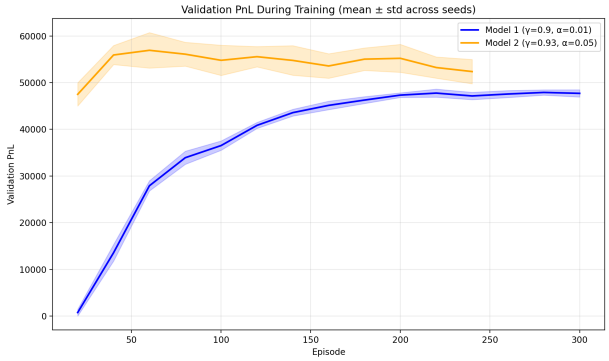
Figure 4a illustrates the agents their learning dynamics. Training rewards start around -€200,000 and improve as ε decays. This is expected because during exploration, random actions accumulate immediate losses. Model 2's less negative training reward (-€50,000 vs. -€150,000 for Model 1 by episode 300) show us a more efficient learning with less costly mistakes.

The validation PnL curves (Figure 4b) show the quality of the true policy. Model 2 its quick increase to €55,000+ within 50 episodes, followed by stable performance, shows an early convergence to a robust strategy. The "persistent" performance gap across all 10 seeds confirms this advantage is due to this specific hyperparameter combination.

Figure 5 provides statistical validation: with 10 independent seeds, Model 2 its 23% improvement over Model 1 is statistically significant. This reproducibility shows us that the learned policy performs reliably regardless of the training randomness.



(a) Training reward (mean ± std).



(b) Validation PnL (mean ± std).

Figure 4: Learning dynamics across 10 seeds for Model 1 and Model 2.

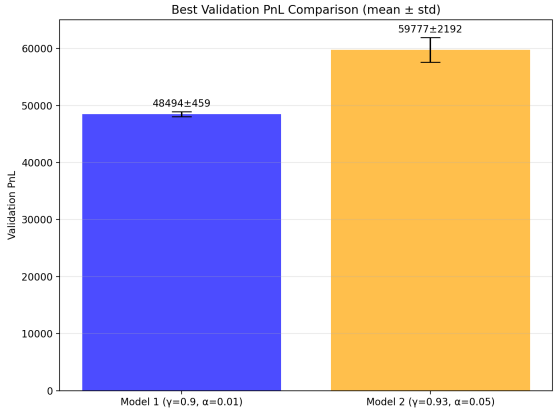
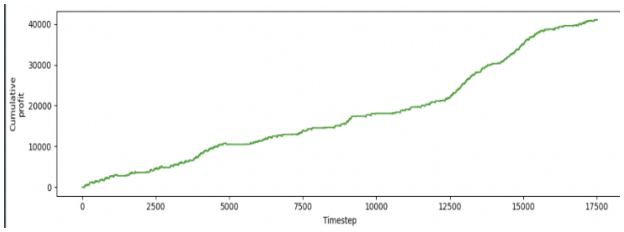


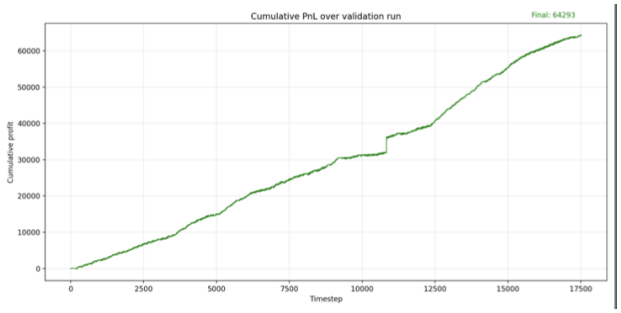
Figure 5: Final validation PnL comparison (mean ± std over 10 seeds). Model 2 achieves €59,777 ± 2,192, outperforming Model 1's €48,494 ± 459 by 23%. Small error bars confirm reproducibility.

3.3 Baseline vs Q-Learning Comparison

Figure 6 compares cumulative validation PnL over time. The baseline threshold policy ends at €40,293 (Figure 6a); the Q-learning model ends at €64,293 (Figure 6b). That is a +€24,000 difference (+59.6%). The Q-learning curve sits above the baseline throughout, with the separation most apparent around timesteps 10,000–12,500.



(a) Baseline threshold policy: €40,293.



(b) Q-learning (final model): €64,293.

Figure 6: Cumulative PnL comparison on validation set. Q-learning achieves 59.6% higher profit than the baseline.

4 Interpretation

4.1 Main findings

The results show that the tabular reinforcement learning agent achieves higher cumulative profit on the validation data than the rule-based baseline. This suggests that the RL agent learns to exploit certain price regimes more effectively, likely through improved timing of buy and sell decisions during high-opportunity periods. To illustrate how the learned policy responds to different market conditions, we include policy heatmaps in the Appendix (Figures 14 and 15) showing agent behavior during contrasting scenarios: fall weekday evenings (expected selling opportunities due to peaking elevated prices) and spring weekend mornings (expected buying opportunities due to lower prices).

4.2 Unexpected learnings

An unexpected finding was that the model combining hyperparameter values that were individually predicted to perform best (Model 1: $\gamma = 0.9$, $\alpha = 0.01$) performed slightly worse than the configuration identified through the two-dimensional grid search (Model 2: $\gamma = 0.93$, $\alpha = 0.05$). This reflects that hyperparameter interactions play an important role in learning dynamics, and that selecting parameters independently may overlook combinations that perform better jointly.

Another unexpected observation was that improving the reward shaping design alone was not enough to correct the imbalance between buy and sell actions. The potential-based reward shaping improved exploration and reduced extreme negative rewards, but it did not fully resolve behavioral issues near storage constraints. This highlighted that state representation, rather than reward magnitude alone, played a dominant role in limiting policy quality.

4.3 Limitations

Despite its improvements, the tabular approach still had many limitations. In particular, the learned policy continues to exhibit a strong imbalance between buy and sell actions, even after applying potential-based reward shaping. Although reward shaping mitigates extreme negative rewards associated with buying and encourages exploration, it cannot fully compensate for structural limitations in the state representation.

Additionally, we observed limitations related to state discretization near the storage boundaries. In the mid-range of storage levels, bins are well visited and learning is stable, which aligns with the intended operating regime. However, near empty or full capacity, coarse discretization fails to distinguish between marginally feasible and infeasible actions. As a result, the agent may attempt to exploit high-price periods without adequately accounting for limited available storage. The 24-hour rollout in Figure 13 (see Appendix) illustrates how the agent continues to select sell actions during high-price periods even after the reservoir has been depleted. The policy heatmaps (Figures 14 and 15) confirm this pattern: in both scenarios, the agent selects actions based primarily on price signals even at the lowest storage bin, where physical constraints should dominate. This boundary insensitivity stems from the coarse discretization, which prevents the agent from distinguishing depleted storage from nearly depleted storage.

While additional feature engineering, including price normalization and temporal features, improved learning stability, these enhancements were insufficient to overcome the limitations imposed by coarse state discretization.

4.4 Future steps

Before moving to more complex function approximation, several targeted modifications could improve the current tabular Q-learning approach. Firstly, the state discretization could be refined near the storage boundaries. Although mid-range storage bins are well visited, coarse bins near empty and full capacity fail to differentiate marginally feasible from infeasible actions. A simple improvement is non-uniform discretization, using finer storage bins near 0% and 100% capacity and coarser bins in the mid-range. This would increase the representational accuracy where feasibility changes abruptly, without significantly increasing the overall state space.

Secondly, the discretization of prices and temporal features could be revisited to improve state visitation. Although quantile-based price bins improve robustness to price spikes, additional structure (e.g., separate bins for extreme peak prices or time-dependent binning) may better capture rare and profitable regimes. In addition, training with shorter segments (e.g., weekly or monthly rollouts) could increase the frequency with which the agent experiences boundary states, improving learning stability in rarely visited regions.

Finally, the table could be extended to include multiple discrete trade sizes (e.g., low/medium/high volumes) rather than always trying to trade the maximum volume. While this increases the action space, it may reduce inefficiencies near low storage levels by allowing partial sell decisions.

Although the above extensions could improve the performance of our tabular Q-learning agent, several limitations remain inherent to the tabular formulation. The tabular agent generalizes across relative price regimes due to normalization and quantile-based features, but its reliance on discrete state representations limits robustness to shifts in price distributions or operating conditions. Moreover, even with refined discretization and additional state features, tabular methods rely on explicit state enumeration and cannot generalize across similar states. This leads to stepwise value functions and persistent sensitivity near rarely visited boundary regions, such as storage limits.

Deep reinforcement learning provides a more flexible alternative by approximating the value function with a neural network operating on continuous state variables. This removes the need for manual binning, enables smoother value estimation, and allows the agent to generalize across nearby states. As a result, deep RL is better suited to capture gradual changes in feasibility and opportunity near physical constraints and to incorporate richer temporal and price features without a combinatorial increase in state space.

5 Conclusion

In this project, we studied the operational control of a pumped-storage hydroelectric dam under uncertain electricity prices using reinforcement learning. We started from a simple, interpretable rule-based baseline, and we demonstrated that a tabular Q-learning agent can significantly improve cumulative profit by learning to exploit relative price regimes and temporal patterns, achieving nearly 60% higher validation profit than the baseline policy. Through extensive feature engineering, reward shaping, and systematic hyperparameter tuning, we identified a robust tabular policy and highlighted the importance of joint hyperparameter interactions. At the same time, our analysis revealed structural limitations of the tabular approach, particularly near storage boundaries where coarse discretization restricts feasibility awareness and generalization. These findings suggest that while tabular reinforcement learning provides a strong and transparent foundation for sequential energy trading problems, further gains are likely to require function approximation methods. In particular, we believe that deep reinforcement learning offers a natural next step to overcome discretization problem, improve generalization across operating regimes, and support more flexible control decisions in realistic energy storage systems.

6 Appendix

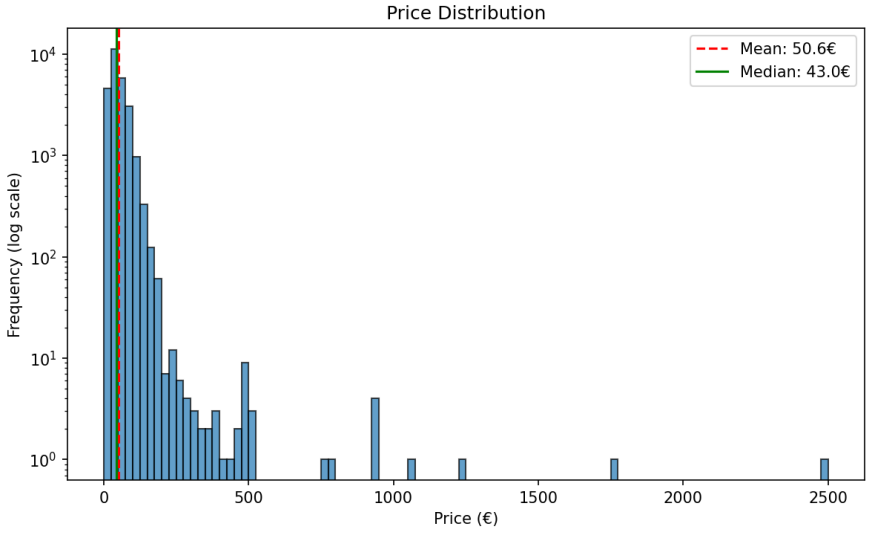


Figure 7: Price distribution, reflecting right-skewed data

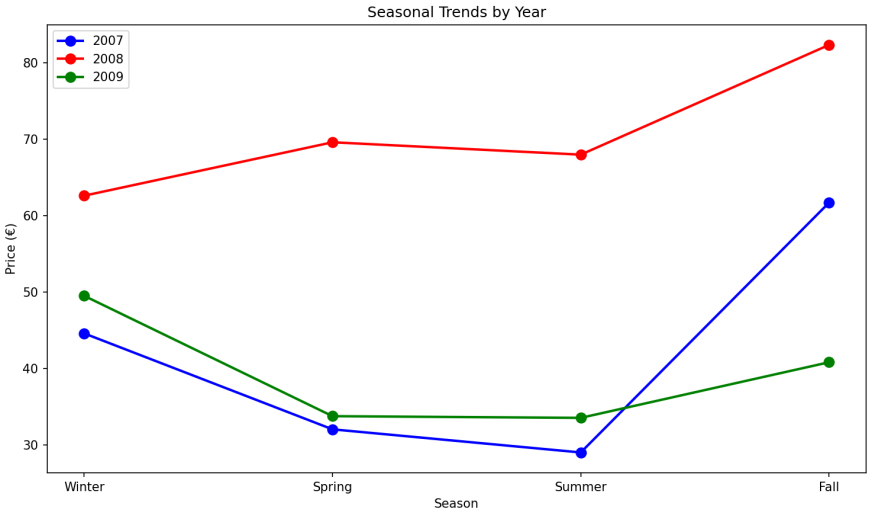


Figure 8: Seasonal price distributions across years, illustrating the shift in price levels during 2008.

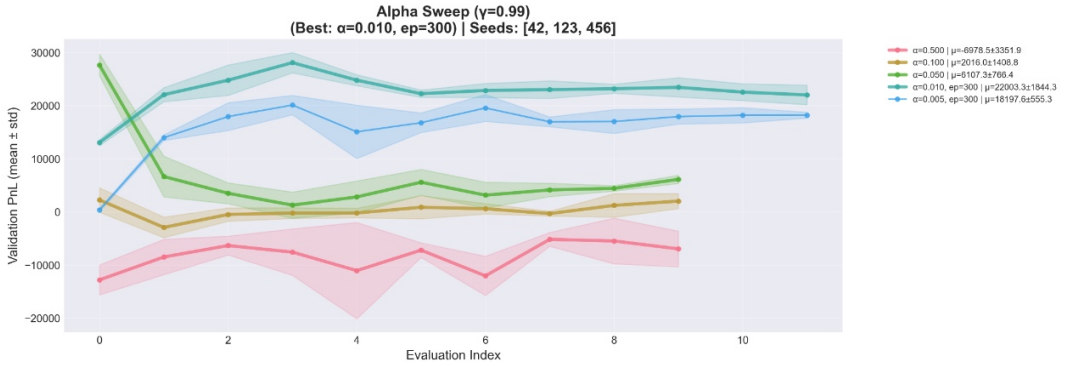


Figure 9: Alpha sweep: validation PnL curves and mean performance across three seeds. Dark green line ($\alpha = 0.010$, 300 episodes) shows best performance.

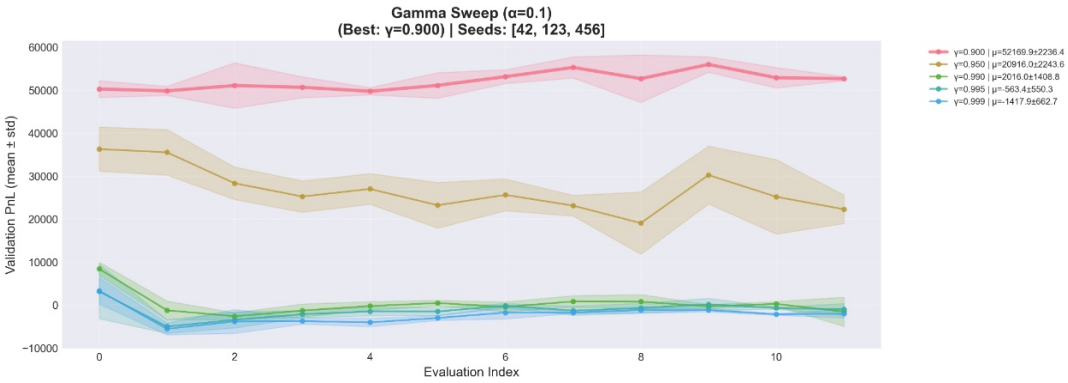


Figure 10: Gamma sweep: validation PnL curves and mean performance across three seeds. Pink line ($\gamma = 0.900$) achieves highest mean final PnL.

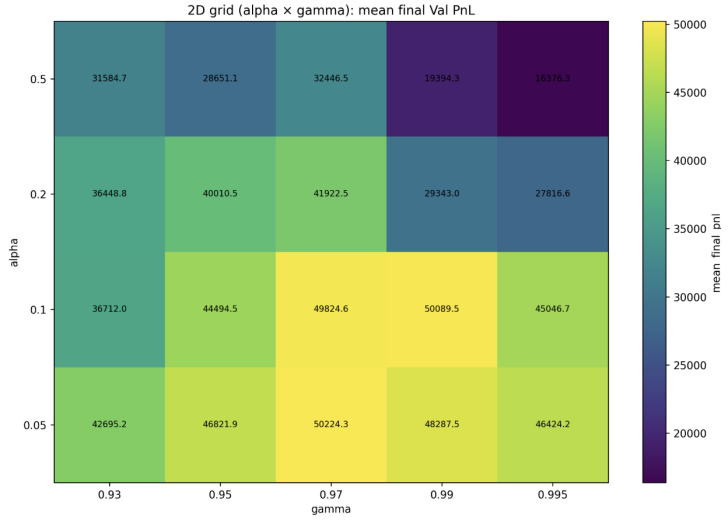


Figure 11: 2D grid search: mean validation PnL across three seeds for all $\alpha \times \gamma$ combinations. Yellow regions indicate high performance; peak at ($\alpha = 0.05, \gamma = 0.97$) achieves €50,224.

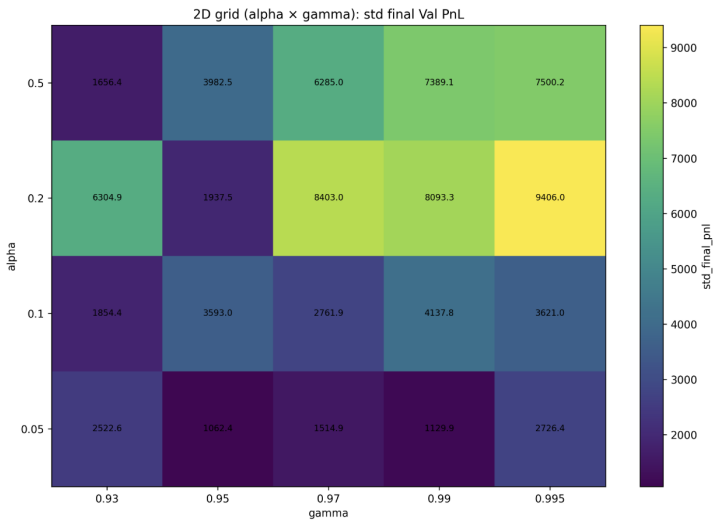


Figure 12: 2D grid search: standard deviation of validation PnL across three seeds. Darker colors indicate lower variance, confirming reproducibility of high-performance regions.

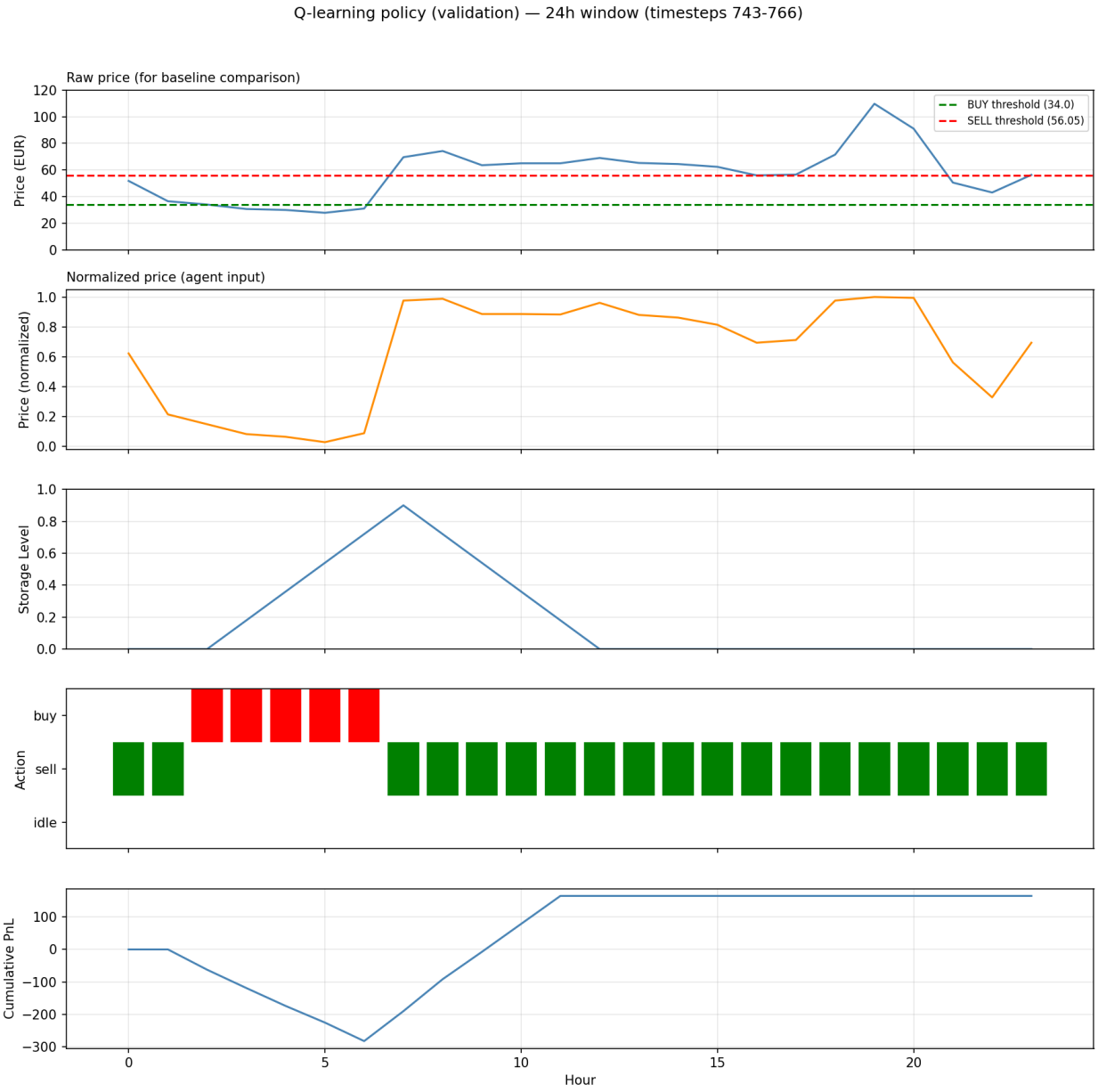


Figure 13: 24-hour rollout illustrating agent action behavior, unable of differentiating nearly empty and empty storage

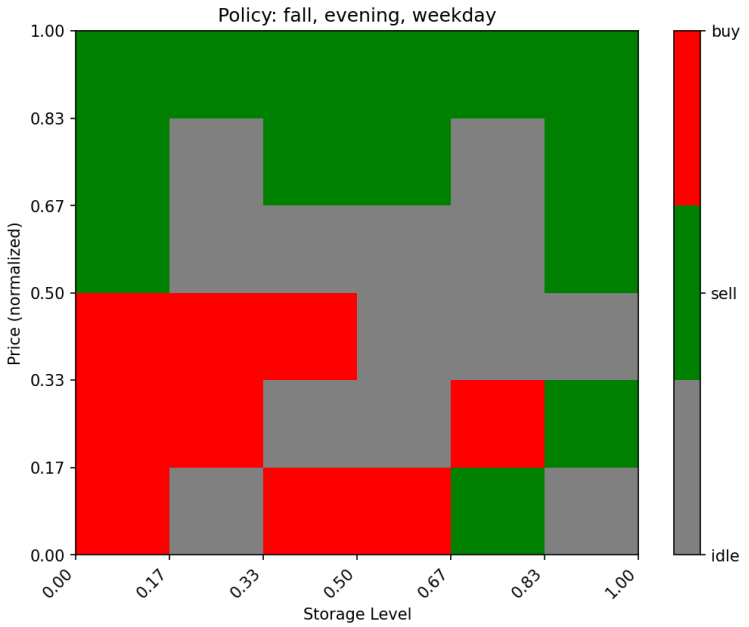


Figure 14: Learned policy for pseudostate (season=Fall, Hourperiod=Evening, DayType=Weekday), reflecting a learned policy of selling when price(normalized) is high regardless of storage state and even buying when price is deemed low

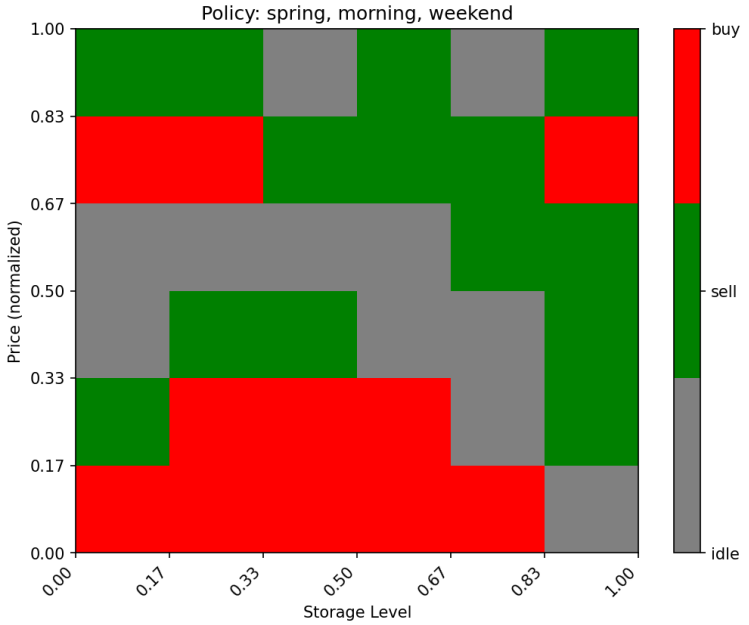


Figure 15: Learned policy for pseudostate (season=Spring, Hourperiod=Morning, DayType=Weekend), reflecting an expected learned policy of buying when price and storage is low.