ChargeTrek: Gamified Learning for Intelligent Optimised EV Charging in Vehicle-to-Grid Systems

Ahmed-Ramzi Houalef[1] Jorge E. Mendoza[2] Florian Delavernhe[1]
Sidi-Mohammed Senouci[1]

[1] Université Bourgogne Franche-Comté, DRIVE, Nevers, France
[2] HEC Montréal, Department of Operations Management and Logistics

Abstract : *Electric vehicles (EVs) equipped with vehicle-to-grid (V2G) capability can reduce energy costs by charging during low-price periods and discharging during high-price periods. While this decision process can be posed as a static optimisation problem, the presence of uncertain electricity prices and strict user mobility constraints turns it into a challenging dynamic optimisation task. ChargeTrek reframes the single-EV home-charging problem as an Atari-style video game in which the agent perceives time, SoC, and price information through a visual grid representation, later solved using learning-based algorithms. Using real CAISO day-ahead and real-time prices, we benchmark reinforcement learning (C51), imitation learning (DAgger), and rule-based baselines. Our results show that the ChargeTrek achieves up to 36% cost reduction over the stepwise planner and 49% over immediate charging. This gamified formulation offers both interpretability and robustness under uncertainty.*

Keywords: Electric vehicles; Vehicle-to-grid; Reinforcement learning; Imitation learning; Gamification; Optimisation.

1 Introduction

Home-charging EVs can exploit electricity price variations by charging when prices are low and discharging when prices peak. Achieving this requires planning under uncertainty: price forecasts contain errors, real-time conditions fluctuate, and the EV must reach a target SoC before departure.

Classical optimisation models rely on deterministic forecasts or static formulations [4][5][6], limiting adaptability. Reinforcement learning (RL) [3][1][2] has shown excellent performance in sequential visual decision-making, especially in Atari environments. ChargeTrek leverages this by transforming EV charging into an interpretable grid-based game.

Our contributions:

- A visual grid-world formulation of EV charging;
- An RGBA encoding conveying price forecasts and forecast errors;
- A comparison of C51 RL, DAgger IL, and optimisation baselines.

Figure 1 illustrates the concept overview.

2 ChargeTrek Environment

2.1 Grid Representation

ChargeTrek models the EV charging process on a two-dimensional discrete grid whose horizontal axis represents time (in 15-minute steps) and vertical axis represents state-of-charge (SoC)

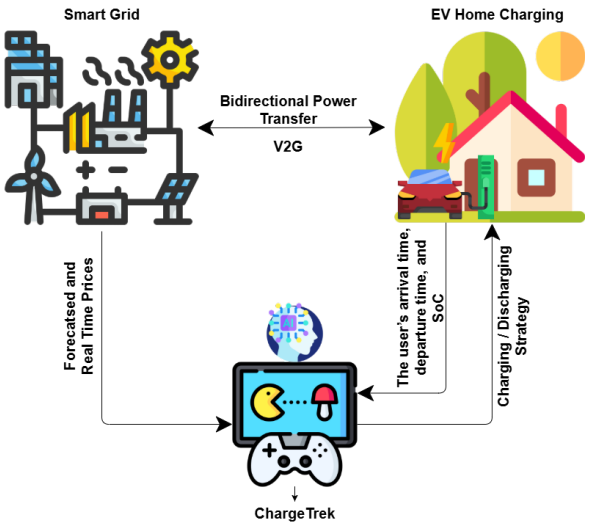

FIG. 1: ChargeTrek concept overview.

levels from 0% to 100%. Each grid cell corresponds to a possible EV state (t, s) and contains visual information that guides the agent's decisions. The agent starts at its arrival time and current SoC and must reach a specific target SoC at the departure time.

Why a grid? The time–SoC grid mirrors real EV dynamics: time always advances, SoC evolves monotonically with charging and discharging, and feasibility can be visualized as reachable or unreachable zones. This makes the problem interpretable both for humans (visual planning) and for learning agents (image-based state representation). It also matches the structure of classic Atari environments, enabling the use of convolutional architectures.

2.2 RGBA Encoding

Each cell is represented using a 4-channel RGBA color vector. This encodes both prices and forecast errors in an interpretable manner:

- **Red (R)** – scaled electricity price. High-price periods appear in saturated red. This encourages avoidance of red areas when charging and creates intuitive negative feedback for the human viewer.
- **Green (G)** – complementary price signal. Defined as $G = 255 - R$, meaning cheap periods appear green. This creates a continuous color gradient from "cheap = green" to "expensive = red," which is easy for convolutional networks to detect.
- **Blue (B)** – sign of forecast error. If the real-time price is higher than forecasted, the cell has $B = 100$. If the forecast overestimated the price, $B = 0$. This allows the agent to distinguish between periods where forecasts tend to be optimistic or pessimistic, making it possible to learn patterns of forecasting bias.
- **Alpha (A)** – magnitude of forecast error. Encodes the absolute difference between forecast and real price, re-scaled to $[0, 255]$. Large errors yield *transparent* cells (low A), making uncertainty visually perceptible. Small errors yield *opaque* cells. This dual encoding (blue=sign, alpha=magnitude) makes forecast uncertainty part of the observable state.

Forecast vs. Real-Time Information At the start of each episode, only day-ahead price forecasts populate the grid (R/G channels). As time progresses and real-time prices are revealed:

- future cells remain forecast-only (no B/A yet),
- past and current cells incorporate B and A based on actual error.

Importantly, ChargeTrek does *not* discard outdated forecasts: the agent sees the forecasted value, the actual value once revealed, and the discrepancy. This allows the agent to learn price-pattern reliability.

Monetary Bar A vertical bar next to the grid accumulates:

- red when charging costs increase,
- green when discharging produces revenue.

This real-time feedback allows the agent to link actions directly to cost consequences through visual signals.

Figure 2 shows an example.

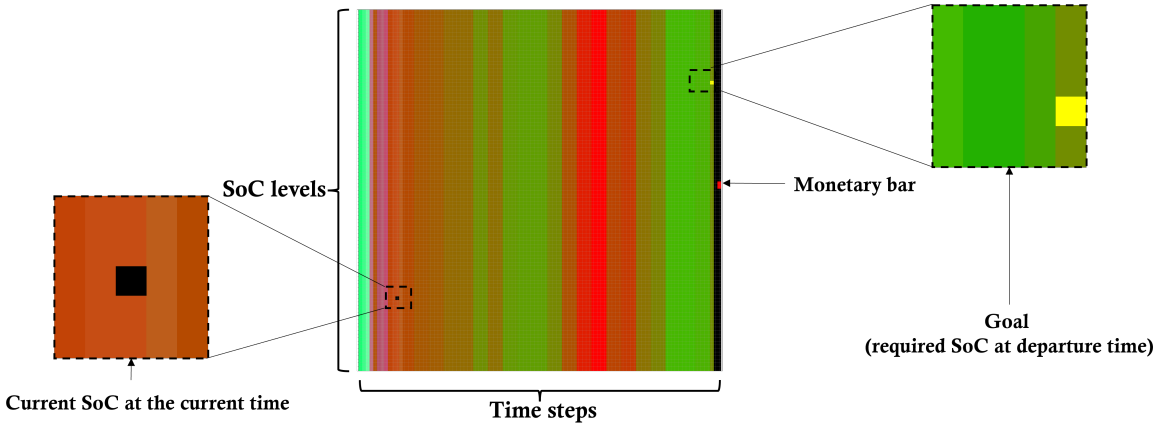

FIG. 2: Environment grid encoding time, SoC, and price uncertainty.

2.3 Data and Battery Model

We use CAISO day-ahead (hourly) and real-time (15-min) LMPs (Jan–May 2025). SoC follows a CC–CV model: 2%/15min up to 80% SoC, then 1%; symmetric tapering for discharge down to 20%.

3 Methodology

3.1 MDP Formulation

ChargeTrek is formulated as a Markov decision process (MDP) $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$ in which each state $s_t \in \mathcal{S}$ is the rendered RGBA grid (including the monetary bar and agent position). The action space is $\mathcal{A} = \{charge,\ discharge,\ idle\}$, and the transition function P is fully determined by the time progression and the battery SoC dynamics. The reward is composed of a monetary component, a feasibility penalty, and a terminal bonus:

$$R_t = R_t^{money} + R_t^{feas} + R_t^{term}.$$

A feasibility penalty R_t^{feas} is applied whenever the chosen action makes the target SoC unreachable before the departure time. At the final step, a terminal reward R_T^{term} is granted if the target SoC is met.

DAgger and the MDP. DAgger operates in *exactly the same MDP*: it observes the same RGBA grid, chooses from the same action set, and follows the same transition dynamics as the RL agent. The key difference is in the training procedure. Instead of learning through temporal-difference updates, DAgger collects state–action pairs by querying an oracle (expert planner) at each visited state. The aggregated dataset is then used to train a policy via standard supervised learning. Thus, DAgger learns a policy that maps the MDP states to actions using imitation rather than value-based optimisation, while remaining fully compatible with the underlying MDP structure.

3.2 Optimisation Baselines

We compare against:

- **Immediate charging**: human-like, no price awareness;
- **Stepwise planner**: recomputes a shortest path via Bellman–Ford using forecast prices;
- **Oracle**: computes an optimal shortest path via Bellman–Ford with perfect foresight of real prices (upper bound).

3.3 Game-to-Graph Transformation

Figure 3 shows how ChargeTrek can be transformed into a time–SoC graph for shortest-path optimisation.

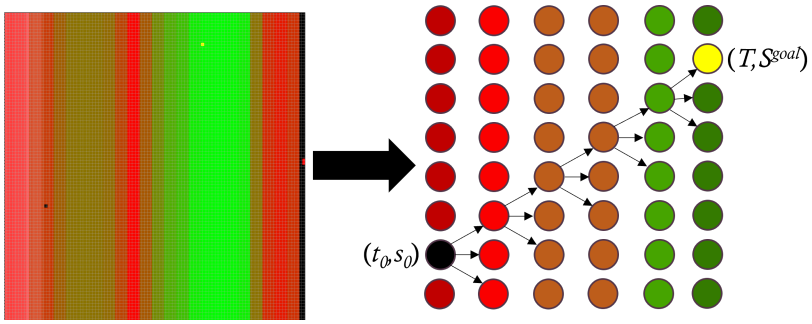

FIG. 3: Game-to-graph structure enabling shortest-path optimisation.

3.4 Learning Approaches

We evaluate two learning-based strategies:

C51 (Distributional RL). The agent predicts categorical value distributions for each action. Learning is done via KL divergence between target and predicted distributions using a replay buffer.

DAgger (Imitation Learning). DAgger iteratively collects trajectories from the current policy, queries the oracle for correct actions, and trains a classifier on aggregated labeled states.

3.5 Hybrid Policy

The hybrid controller executes the learned action if it keeps the goal feasible; otherwise, it temporarily switches to the stepwise planner.

4 Experiments

4.1 Setup

We use 15 diverse user profiles with varying arrival, departure and SoC targets. CAISO days are split 80/20 for training/testing. Agents train for 2M steps using burst training.

Metrics:

- **SoC gap**: achieved vs. required SoC;
- **Gain**: relative improvement over a baseline;

4.2 Results

The evaluation considers both training and test episodes in order to measure how well each agent generalises beyond the price patterns seen during learning. Violin plots are used to jointly visualise two complementary aspects: **cost gain** and **SoC-tracking reliability**. The *colour* of each violin encodes the gain of the agent relative to a selected baseline (oracle, stepwise, or immediate). Greener shades indicate higher positive gain, meaning the agent achieves greater cost savings than the baseline; redder shades indicate negative gain, where the agent performs worse; and whiter colours correspond to zero gain, meaning the agent behaves approximately as well as the baseline. This colour information is shown for both training and test distributions, allowing us to assess not only performance but also generalisation.

The *shape* of each violin reflects the distribution of the SoC gap across all episodes in the corresponding split (train or test). A narrow, centred violin around zero indicates that the agent consistently reaches the required SoC before departure. Wider or off-centred distributions reveal unreliable charging behaviour or frequent violations of user mobility constraints. By combining colour (gain) and distribution shape (SoC feasibility), these plots provide a compact visual summary of both economic and operational performance. For hybrid agents, the SoC-gap distribution is included for symmetry, but it effectively collapses to zero in all cases since the fallback ensures the required SoC is always achieved.

C51 improves over training but shows noticeable SoC variability and inconsistent cost gains. This is confirmed in Fig. 4, where the C51 SoC distribution remains wide ($\pm 40\%$), especially on the test set, showing a lack of reliability.

DAgger is significantly more stable and cost-efficient:

- pure DAgger: +36% price gain vs stepwise, +49% vs immediate,
- hybrid DAgger: perfect SoC satisfaction and +22% price gain vs stepwise, +34% vs immediate.

Figure 5 shows the SoC gap distribution for DAgger. The values are centered around 0 where the distribution remains between ($\pm 10\%$), demonstrating better control and very low risk of undercharging or overcharging. Overall, DAgger performs remarkably close to the oracle on many episodes despite having no access to future prices. The hybrid DAgger variant further strengthens this behaviour: it consistently reaches the required SoC in every episode and achieves a good price gains, combining the efficiency of imitation learning with the reliability of the stepwise fallback mechanism.

5 Conclusion

ChargeTrek reframes EV charging as a visual grid game. C51 RL provides moderate improvements but struggles with SoC stability, whereas DAgger IL consistently delivers strong gains while satisfying mobility constraints. Hybrid DAgger offers the best trade-off between cost, feasibility and robustness.

Future work includes multi-EV extensions, richer user modelling and larger datasets.

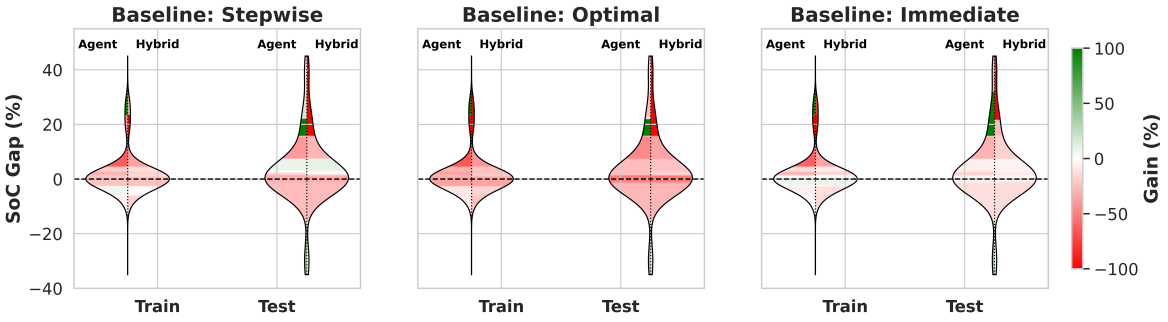

FIG. 4: SoC distribution for the C51 agent across training and test sets. The wide violin shape indicates large variability and unstable performance, confirming that C51 struggles to maintain consistent cost savings.

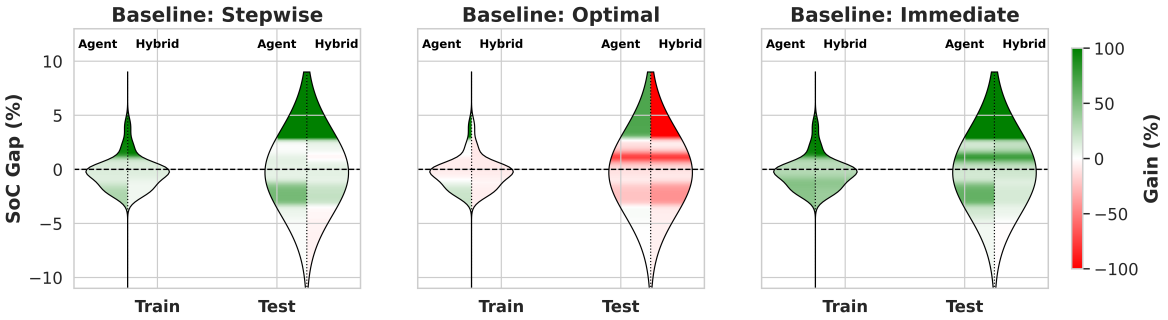

FIG. 5: SoC gap distribution for the DAgger agent. The narrow, centered violin demonstrates highly stable SoC tracking and reliable satisfaction of user mobility constraints.

References

[1] Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *International conference on machine learning*, pages 449–458. PMLR, 2017.

[2] Nicholas D Kullman, Nikita Dudorov, Martin Cousineau, Jorge E Mendoza, and Justin C Goodson. Gamifying the vehicle routing problem with stochastic requests. *INFORMS Journal on Computing*, 2025.

[3] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei Rusu, Joel Veness, Marc Bellemare, Alex Graves, Martin Riedmiller, Andreas 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*, 518:529–33, 02 2015.

[4] Sagar Mody and Thomas Steffen. Optimal charging of electric vehicles using a stochastic dynamic programming model and price prediction. 1 2015.

[5] Marina González Vayá, Luis Briones Roselló, and Göran Andersson. Optimal bidding of plug-in electric vehicles in a market-based control setup. In *2014 Power Systems Computation Conference*, pages 1–8, 2014.

[6] Arjun Visakh and M. Selvan. Charging-cost minimization of electric vehicles and its impact on the distribution network. pages 1–5, 12 2021.