EV2Gym Driveway

Nicholas Khorasani and David Wu, nck2130 and dsw2162

May 11, 2025

1 Abstract

As the adoption of electric vehicles (EVs) increases, so does their potential to serve as distributed energy assets through vehicle-to-grid (V2G) integration. Coordinated charging and discharging of EVs can reduce consumer electricity costs and lower peak grid load. While centralized control achieves strong performance, it raises concerns around privacy and scalability. In contrast, decentralized approaches offer better privacy but often underperform centralized algorithms. This project explores a middle ground: semi-centralized coordination, where EVs are grouped into clusters of varying sizes. We introduce a new simulation environment, EV2Gym-Driveway, based on the existing EV2Gym simulator, and evaluate a Proximal Policy Optimization (PPO) agent across different coordination group sizes. We benchmark PPO's performance against a naive heuristic baseline algorithm called Charge As Fast As Possible (CAFAP) and evaluate cost savings and peak load reduction. Our results show that PPO significantly reduces electricity costs and peak load when coordinating three vehicles. However, as we scale to larger groups (e.g., 5 or 10 EVs), PPO fails to consistently outperform CAFAP. We attribute this to high sensitivity to both hyperparameter configurations and reward function design, which may require retuning for each group size.

2 Introduction

As the impacts of climate change become increasingly apparent, reducing humanity's reliance onn fossil fuels has become a global priority. Renewable energy sources such as solar and wind are increasingly integrated into power systems, offering a cleaner alternative to fossil fuels. However, these sources are inherently intermittent. Solar panels do not generate power at night, and wind turbines depend on weather conditions. Therefore, even in regions with many renewable energy sources, we still rely on fossil-fuel-based peaker power plants to meet energy demands.

Large-scale energy storage is a promising solution to the inconsistent availability of renewable energy. The goal is to store excess energy when supply exceeds demand and release it back to the grid when demand rises or generation falls. However, deploying grid-scale batteries and other dedicated storage systems requires significant capital investment, slowing widespread adoption. Fortunately, the rapid growth of the adoption of electric vehicle (EV) presents a unique opportunity. EVs already have substantial onboard battery capacity and are idle for most of the day. When parked and connected to the grid, these vehicles can serve as distributed energy storage units—charging during off-peak hours and discharging power back to the grid when demand spikes.

The idea of using electric vehicles as distributed energy storage devices is known as Vehicle-to-Grid (V2G). In a V2G-enabled system, EVs can charge during low demand or high renewable generation times and discharge electricity back into the grid during peak hours. This coordinated behavior can help flatten demand curves, reduce reliance on peaker plants, and even provide cost savings to consumers.

In this project, we investigate how reinforcement learning can coordinate EV charging and discharging across a neighborhood. Our primary goals are to reduce electricity costs and peak

charging load. We explore how different levels of coordination affect system performance using our EV2Gym-Driveway simulator, which we built on top of the EV2Gym simulator (SOURCE FOR EV2Gym). By comparing a PPO-based reinforcement learning agent against a naive charging policy, we aim to evaluate the trade-offs between coordination complexity and overall efficiency.

3 Challenges with V2G

Several practical hurdles stand in the way of making Vehicle-to-Grid (V2G) systems work at scale. Some obvious barriers include setting up the physical infrastructure (installing sensors to collect real-time data), upgrading charging stations to support both charging and discharging, and ensuring that transformers can handle the extra load electric vehicles will put on the grid.

In this project, we focus on two specific challenges that are just as important but often overlooked: **privacy** and **scalability**.

3.1 Privacy

The first challenge is privacy. To effectively coordinate EV charging and discharging algorithms requires access to information such as departure and arrival times, daily driving patterns, and battery state. This is very personal data, so many users are hesitant to share such data with a central authority. However, we hypothesize that privacy concerns may be mitigated if coordination occurs within smaller, localized groups. For example, users may be more willing to share data with a small set of neighbors (e.g., 3–10 households) than with a utility company or a centralized aggregator. This hypothesis is why we are investigating small-scale EV coordination in this project.

3.2 Scalability

The second challenge is scalability. As the number of EVs increases, coordinating them all under a single centralized algorithm becomes very computationally expensive. This is because the joint state and action spaces grow exponentially with the number of EVs. Therefore, many reinforcement learning algorithms could not make coordination decisions fast enough to be practically useful.

In this project, we're looking for a coordination "sweet spot"—small enough to be computationally feasible and respect privacy, but large enough to actually make a difference.

4 Related Works

Although none have attained significant widespread use as of yet, many people have extensively researched systems for central coordination of charging EVs, in order to prepare for an influx of electric vehicles into the greater power grid ecosystem. Among the many possible paradigms to tackle this problem, reinforcement learning is particularly suited towards optimizing coordinated EV charging, not least due to the readily apparent mappings into agent and action spaces. We mainly referenced two prior works that deal with reinforcement learning for this kind of EV charging.

4.1 Coordinated Algorithms

Reinforcement learning for electric vehicle applications in power systems: A critical review by Qiu *et al.* introduces a range of reinforcement learning algorithms suitable for electric vehicle coordination tasks, including value-based methods like Deep Q-Networks (DQN) and policy-based approaches such as Proximal Policy Optimization (PPO). Among these is the distinction between centralized and decentralized training and centralized and decentralized execution. The frameworks of centralized training with centralized execution, centralized training with decentralized execution, and decentralized training with decentralized execution are all widely

used for multi-agent reinforcement learning algorithms. This is in contrast to single-agent reinforcement learning algorithms, which include the aforementioned DQN and PPO.

4.2 EV Simulators

On the other hand, Orfanoudakis *et al.* present a detailed overview of *EV2Gym*, a sophisticated and flexible simulation environment for EV smart charging, built using Python's OpenAI Gymnasium library. Additionally, they alluded to previous simulators such as V2GSim and EVLibSim, but concluded that they were each deficient in one way or another. For example, V2GSim is not open source. Conversely, EV2Gym is very flexible in that it supports various customizations, including diversity in EV characteristics and specifications, as well as many choices of algorithms, reinforcement learning, or otherwise. Interestingly, "RL approaches appear to be more conservative in approaching the transformer power limits" [2, p. 9]. Our project, EV2Gym_Driveway, adapts this simulator to model a neighbourhood with houses, each with a single EV and a single charging station.

5 Methodology

5.1 EV2Gym Driveway Simulator

As stated above, our EV2Gym_Driveway repository is essentially an adaptation of EV2Gym where the focus shifts to individual households, each with their own EV, and the simulator measures the interaction between those households, which can vary based on the number of households in the group.

5.1.1 EV2Gym_Driveway Class Structure

The basic building blocks are ev.py and Household.py. The ev.py class represents an EV entity and contains basic properties such as the current battery and current state of charge (SoC), as well as intrinsic methods to charge and discharge the EV. The Household.py class represents a single household. In this version of the simulator specifically, each household possesses exactly one EV and is associated with exactly one charging station. Thus, each EV-Household pair can be considered as one building block for the greater environment of the simulation.

Two other important classes are ev_charger.py and transformer.py. The ev_charger.py class represents one charging station. The charging station contains a specified number of ports and tracks relevant statistics, including flow of profit between it and visiting EVs, plus user satisfaction, which is a component of some reward functions. The transformer.py class represents one transformer entity, which dictates the maximum and minimum power limits for each charging station under its jurisdiction. Care must be taken not to overload transformers as that breaks constraints and can even prematurely end the simulation.

Each of these four classes contains init(), reset(), and step() functions, which indicate that they are part of a larger working that progresses through each time step of one run of the simulation. Speaking of which, that central class of the entire repository is ev2gym_driveway_env.py, which represents the environment of the simulation. It contains the master init(), reset(), and step() functions, the last of which is synced with the timestep of the greater simulation. Additionally, it tracks global statistics for the simulation, which is important for later stage analysis. Finally, this class gets imported and called by the top level Jupyter notebooks that read in the preparatory config files and run the simulation, as part of one of the reinforcement learning algorithms.

There are other minor ancillary classes, such as loaders.py and statistics.py in the utilities directory, reward.py for the reward function and state.py for state space creation in the rl_agent directory, and heuristics.py in the baselines directory, as well as a multitude of other classes, but the five classes listed above are of cardinal importance.

5.1.2 Simulator Mechanics

In a nutshell, the simulator can be summed up by the following pseudocode:

For each EV:

*Init() by sampling **weekly trip profile** from NHTS*

Repeat for each time step:

Check if EV is at home

If EV is at home:

Decide whether to charge, discharge, or not

Else, EV departed:

Make sure to disconnect until EV returns

Track statistics such as energy use, charging state etc.

Different reinforcement learning algorithms can use this simulator in different ways and time frames to suit their specific needs.

5.2 EV Charging Coordination Algorithms

5.2.1 Baseline: Charge As Fast As Possible

This is one of the baseline heuristics, best described as greedy. The charging station charges all connected EVs as much and as immediately as possible, up to the maximum allowed by the transformer. The other models are compared against this "naive" heuristic of Charge as Fast as Possible, hereafter referred to as CAFAP.

5.2.2 Reinforcement Learning (RL)

State Space: This is defined as the current state of the environment, which is normally represented by a state vector. For EV2Gym_Driveway this would include battery level, charge and discharge prices, time of day etc.

Action Space: This is defined as the set of all possible actions an agent can take when in the current state space. For a single EV this includes charging, discharging, or doing nothing.

Reward Function: This is defined as the goal for the reinforcement learning algorithm to maximize. Here, it is a function of money earned and spent via discharging and charging, plus some optional parameters like user satisfaction, which depends on the percentage charged of an EV, and invalid action punishment.

RL Algorithm: Proximal Policy Learning (PPO) This is a policy based algorithm similar to Policy Gradient [1, p. 6]. For this project, we incorporated the implementation of PPO from the StableBaselines3 module.

RL Algorithm: Deep Q Network (DQN) This is a value based algorithm that is essentially an extension of traditional Q-learning to employ deep neural networks. This allows it to handle continuous state spaces, but action spaces must still be discretized.

6 Results

6.1 PPO

We evaluated our PPO algorithm on coordination group sizes of 3, 5, and 10 households. We attempted to evaluate PPO on larger coordination sizes (15 and 20 households), but the machines we ran our code on did not have enough memory. For each group size, we compared

the PPO agent's performance to a naive baseline (Charge As Fast As Possible, or CAFAP). We assessed two primary metrics: household cost savings and load volatility.

6.1.1 Cost Savings

Table 1 presents the percentage reduction in electricity costs achieved by PPO relative to CAFAP across different coordination group sizes over a one-week simulation. With a coordination size of 3, PPO successfully reduced household electricity costs by 12.46%. However, when the same reward function and hyperparameters were applied to larger group sizes (5 and 10), PPO's performance degraded significantly, resulting in higher costs than the naive baseline. This means that the PPO agent is sensitive to changes in the coordination group size and suggests that careful tuning of the reward function and hyperparameters is essential.

Table 1: Percentage Cost Savings by Group Size (PPO vs. CAFAP)

Group Size	PPO Cost Relative to CAFAP Cost (%)
3	-12.46
5	139.19
10	108.33

6.1.2 Grid Load

To evaluate how well PPO reduces peak load, we collected two key metrics: the standard deviation and the peak-to-average ratio. Lower values in both metrics indicate smoother and more grid-friendly charging. Table 2 shows the results across coordination group sizes. Like with cost, PPO reduced the standard deviation and peak-to-average for the case of coordinating three EVs in comparison to CAFAP. But it failed to beat CAFAP for the larger coordination groups.

Table 2: Grid Load Metrics Across Coordination Sizes

Group Size	Metric	CAFAP	PPO	Difference
3	Std. Dev.	1.85	1.61	-12.99
	Peak/Avg Ratio	28.12	25.79	-8.29
5	Std. Dev.	3.03	5.47	80.27
	Peak/Avg Ratio	26.53	26.53	0.00
10	Std. Dev.	3.76	6.22	65.55
	Peak/Avg Ratio	14.62	17.79	21.65

6.1.3 Visualizations

We present key visualizations of PPO's behavior in the 3-vehicle scenario, which achieved better performance than the CAFAP baseline. Figure 1 shows the training reward curve over time, showing that the PPO algorithm converged on a strategy over time. Figure 2 displays cost savings per vehicle over one week of simulation when PPO coordinates charging. From Figure 2, you can see some EVs saved money (EVs 0 and 2) while EV 1 did not. But overall, the group saved 12.46%.

6.2 DQN

Figure 3 shows the training curve for DQN. Each increment represents one training episode, in other words from one reset to the next reset, and the metric used is the total reward across the entire episode. As indicated, apart from an upward fluke at the very beginning and one downward fluke towards the end of training, the curve shows a significant improvement, although it is unstable between episodes.

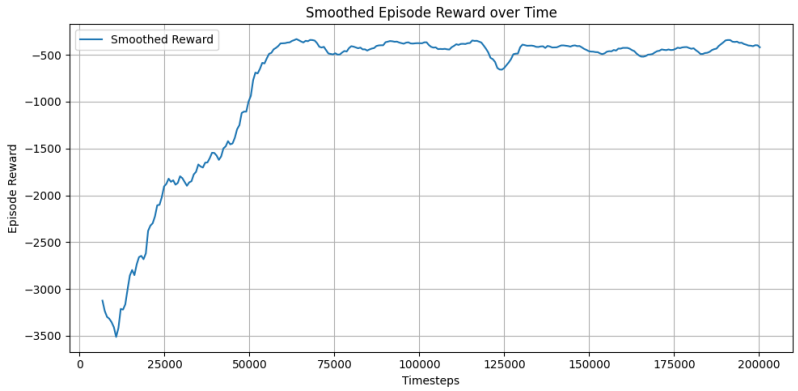

Figure 1: Training reward curve for PPO in the 3-vehicle case.

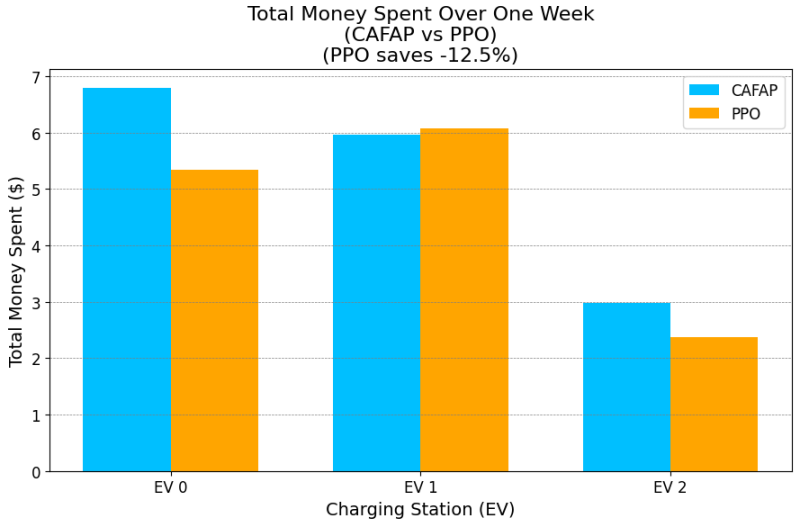

Figure 2: Cost per EV over simulation episodes (3-vehicle case).

Finally, Figure 4 shows the distribution of power usage for DQN as compared to CAFAP. In summary, peak power usage for DQN is much lower than for CAFAP, and the charging schedules for DQN and CAFAP do not always line up.

7 Conclusion and Discussion

PPO showed significant improvement in costs saved and peak load reduction when compared to CAFAP, but only for the 3-vehicle case. We attribute this to the fact that we tuned the hyperparameters of our algorithm for the 3-vehicle case, and it did not extend to larger coordination sizes. That being said, we expect that with better hyperparameter tuning, PPO could result in both cost savings and peak load reduction for coordination groups of size 5 and 10.

Despite the limitations of PPO, it is still better suited for this problem than DQN because, as a value based approach, DQN requires keeping track of too many moving parts, with the policy being a separate neural network from everything else. In addition, the requirement of a discrete action space is a huge point against DQN, because too small of an action space yields inaccurate results, but too large of an action space takes too long to train, and increasing the action space yields diminishing returns as the agent does not have a lot of time to try out all the possible discrete actions. Another counterpoint is that reinforcement learning using DQN is inherently very noisy. The success of each run fluctuates wildly depending on the agent's luck

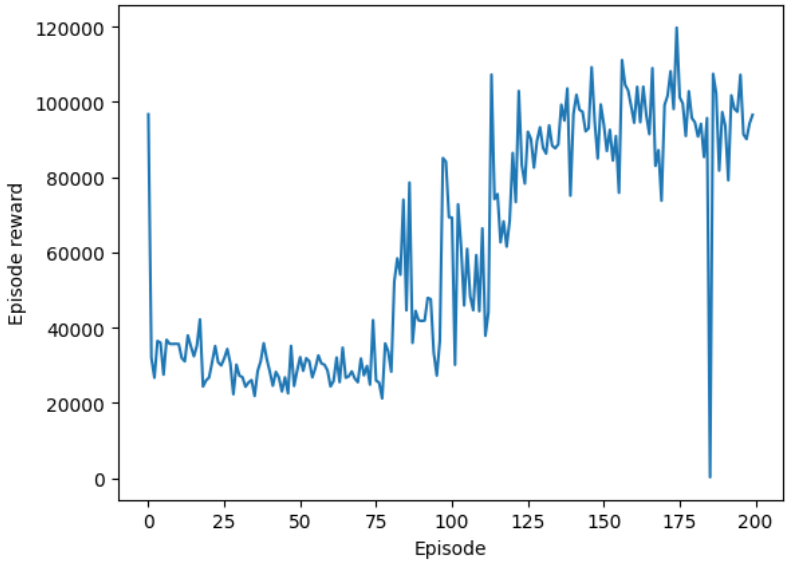

Figure 3: DQN Training Curve

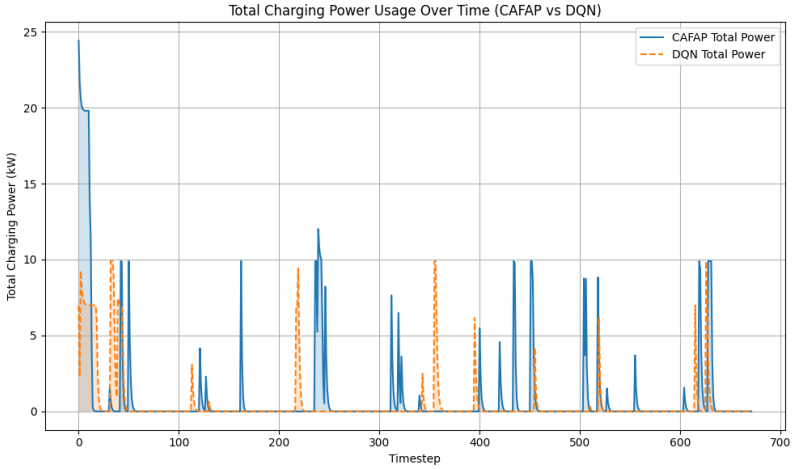

Figure 4: DQN Total Power Usage

in the exploration phase.

8 Limitations and Future Work

Battery Degradation Model: One of the limitations of the EV2Gym-Driveway simulator is that we could not integrate a battery degradation model due to time constraints. Because a V2G system would require lots of charging and discharging, the battery will also suffer from additional degradation. Therefore, a practical V2G reinforcement learning system should consider battery degradation in its reward function to ensure that the EVs functionality is not sacrificed by participating in a V2G system.

Hyperparameter Tuning: In this project, we manually evaluated many different reward functions and hyperparameters. During this process, we evaluated factors representing *User Satisfaction*, *Money Spent and Earned Charging*, *Grid Load*. In the future, it would be valuable to set up our environment in a way that we could avoid manually tuning a large number of hyperparameters.

Including Grid Load Factor In Reward Function: Our final reward function only

included terms for *User Satisfaction*, *Money Spent and Earned Charging* because we were unable to find a formula that led to stable training when including a factor that represents *grid load*. This is a major limitation since one of the goals of this project was to reduce peak grid load. Future V2G RL systems based on this simulator should include a grid load factor in the reward function.

Lifting the one EV per household restriction: In the EV2Gym-Driveway simulator, there is a strict one EV per household limit. However, many homes have multiple EVs so lifting this restriction would better represent real life.

9 References

[1] Qiu, D., et al. "Reinforcement learning for electric vehicle applications in power systems:A critical review," in Renewable and Sustainable Energy Reviews, vol. 173, pp. 113052, 2023.

[2] Orfanoudakis, S., et al. "EV2Gym: A Flexible V2G Simulator for EV Smart Charging Research and Benchmarking," in IEEE Transactions on Intelligent Transportation Systems, vol. 26, no. 2, pp. 2410–2421, 2025.