Chargax: A JAX Accelerated EV Charging Simulator

Anonymous authors
Paper under double-blind review

Keywords: Jax, EV Charging, Gym Environment, Reinforcement Learning, Benchmarking

Summary

Deep Reinforcement Learning can play a key role in addressing sustainable energy challenges. For instance, many grid systems are heavily congested, highlighting the urgent need to enhance operational efficiency. However, reinforcement learning approaches have traditionally been slow due to the high sample complexity and expensive simulation requirements. While recent works have effectively used GPUs to accelerate data generation by converting environments to JAX, these works have largely focussed on classical toy problems. This paper introduces Chargax, a JAX-based environment for realistic simulation of electric vehicle charging stations designed for accelerated training of RL agents. We validate our environment in a variety of scenarios based on real data, comparing reinforcement learning agents against baselines. Chargax delivers substantial computational performance improvements of over 100x-1000x over existing environments. Additionally, Chargax' modular architecture enables the representation of diverse real-world charging station configurations.

Contribution(s)

(i) This paper presents Chargax, an open-source EV charging environment written in JAX
Context: Chargax could be used as a high-performance test bed for reinforcement learning benchmarking, or to develop better control algorithms for EV charging.

(ii) Comparisons in performance are made with previously existing EV simulators for RL that demonstrate Chargax decreases training times by a factor of 100x or more.
Context: Prior work established EV charging simulators for RL that did no leverage the GPU

(iii) We perform additional experiments validating reinforcement learning training in a variety of scenarios, data distributions shifts, and reward objectives.
Context: None

(iv) We create an explicit split in the state space which highlights the interchangeable parts in the Chargax environment. This modularity allows representation of diverse real-world charging station configurations and scenarios.
Context: Prior work often used this split implicitly, and allow for less customisability

Chargax: A JAX Accelerated EV Charging Simulator

Anonymous authors
Paper under double-blind review

Abstract

Deep Reinforcement Learning can play a key role in addressing sustainable energy challenges. For instance, many grid systems are heavily congested, highlighting the urgent need to enhance operational efficiency. However, reinforcement learning approaches have traditionally been slow due to the high sample complexity and expensive simulation requirements. While recent works have effectively used GPUs to accelerate data generation by converting environments to JAX, these works have largely focussed on classical toy problems. This paper introduces Chargax, a JAX-based environment for realistic simulation of electric vehicle charging stations designed for accelerated training of RL agents. We validate our environment in a variety of scenarios based on real data, comparing reinforcement learning agents against baselines. Chargax delivers substantial computational performance improvements of over 100x-1000x over existing environments. Additionally, Chargax' modular architecture enables the representation of diverse real-world charging station configurations.[1]

1 Introduction

Deep Reinforcement Learning (RL) can approximate optimal policies for difficult decision problems that are impossible to solve with traditional mathematical methods. Such problems occur frequently in sustainable energy challenges such as operation of windfarms (Fernandez-Gauna et al., 2022), electric vehicle charging (Rehman et al., 2024), and nuclear fusion reactors (Seo et al., 2024). While RL has achieved successful solutions to these challenges, further development of RL algorithms hinges on the availability of realistic simulation environments and benchmarks (Ponse et al., 2024).

Unfortunately, reinforcement learning is notoriously sample-inefficient (Yarats et al., 2020; Kaiser et al., 2024). It often requires many environments samples which are slow and possibly expensive to generate. These simulations have often been running on the CPU – disallowing RL researchers from truly harvesting the potential scale-up of GPUs that other machine learning fields have been enjoying (Scarfe et al., 2025). To this end, the development of RL environments using JAX (Bradbury et al., 2018) has recently gained increasing attention (Freeman et al., 2021; Lange, 2022; Pignatelli et al., 2024; Bonnet et al., 2024). However, current implementations remain largely confined to simplified toy problems, highlighting a significant gap in real-world applications utilizing JAX.

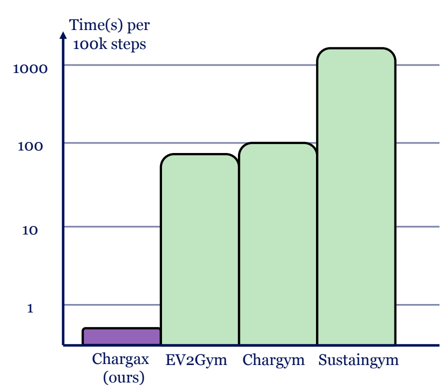


Figure 1: Comparison between Chargax and prior EV Gym Environments in time in seconds to complete 100k training steps using a PPO agent. See Table 2 for a more complete overview.

[1] Available on GitHub at https://github.com/anonymous-submission/anonymous-submission

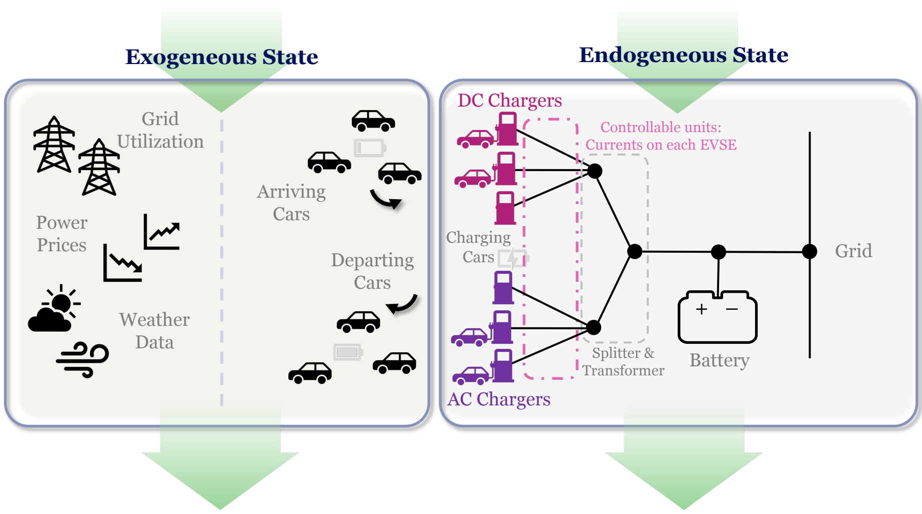


Figure 2: An overview of the Chargax environment. The *endogenous state* describes the state variables that are influenced directly by the agent. The *exogenous state* evolves via, agent-independent, predefined time series data.

Contribution In this work, we aim to bridge this gap by introducing, to the best of our knowledge, the first reinforcement learning environment for EV charging implemented in JAX.

- Our environment, *Chargax*, achieves a significant speedup of 100x-1000x compared to existing environments for EV charging (Yeh et al., 2024; Orfanoudakis et al., 2024; Karatzinis et al., 2022). This lowers training times from hours or even days to mere minutes – allowing for orders of magnitude more experiments (see Figure 1).
- Chargax extends the generalisability of existing frameworks. As highlighted in a recent survey (Alaee et al., 2023), optimising electric vehicle charging strategies involves a diverse set of potential objectives. We demonstrate that many of these objectives can be addressed within a single simulation framework by ensuring sufficient flexibility.
- Chargax can function as a high-performance test bed for reinforcement learning benchmarking on real-world applications. Empirically, we demonstrate how RL agents are able to outperform baselines and allow for flexible goals such as user satisfaction. We open source *Chargax*[1] for the wider community to experiment with.

Chargax is equipped with predefined datasets, reward functions, and charging station architectures for various scenarios. Moreover, all components are fully customizable, enabling researchers to tailor the environment to specific requirements, thereby facilitating efficient and adaptable RL-based solutions for EV charging optimization.

2 Related Work

Prior work in EV charging includes the gym environments Sustaingym (Yeh et al., 2024) (based on (Lee et al., 2020b)), Chargym (Karatzinis et al., 2022), and the more recently released EV2Gym (Orfanoudakis et al., 2024). Compared to (Yeh et al., 2024; Lee et al., 2020b; Karatzinis et al., 2022) our framework provides additional flexibility for customer and car profiles. In addition, the architecture of the Charging station as well as the selection of scenarios. Compared to (Orfanoudakis et al., 2024), which also prioritises flexibility, our approach features a more streamlined state and architecture representation. To the best of our knowledge, Chargax is the only Gym-like environment that includes local car and price data across multiple regions. Furthermore, Chargax is orders of magnitude faster and in turn allows for large scale experiments on the GPU (See Figure 1). Apart from these Gym-like simulators, there exist a history of EV charging simulators (Saxena, 2013; Rigas et al., 2018; Balogun et al., 2023; Cañigueral, 2023).

In recent years, many classical Gym environments have been reimplemented in JAX. We direct the reader to the following non-exhaustive list (Freeman et al., 2021; Lange, 2022; Nikulin et al., 2023; Rutherford et al., 2023; Koyamada et al., 2023; Pignatelli et al., 2024; Bonnet et al., 2024). These implementations have largely been reimplementations of classical toy problems, highlighting a gap in environments modelling real-world problems.

3 Preliminaries

Markov Decision Process

Formally, an environment is represented as a Markov Decision Process (MDP; Sutton & Barto 2018) defined by a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p_0, p, r, \gamma)$. Here, $\mathcal{S}$ is a state space, $\mathcal{A}$ is a action space, $p_0 \in \Delta(\mathcal{S})$[2] is the initial state distribution, $p(\cdot|s, a) \in \Delta(\mathcal{S})$ is the probabilistic transition function, $r(s, a, s')$ denotes the reward function and $\gamma \in [0, 1)$ is the discount factor. In the next section (4), we provide a detailed discussion of the motivation behind the choices for each MDP component and formally define these quantities used within the framework.

JAX

JAX is a Python library aimed at accelerator-orientated programming with a NumPy interface (Bradbury et al., 2018). It offers function transformations to perform, for example, just-in-time-compilation, vectorization, and differentiation. Although JAX is a common foundation for deep learning frameworks (Heek et al., 2024; Kidger & Garcia, 2021), its just-in-time compilation transformation allows users to easily run plain Python code on accelerators such as GPUs and TPUs. Although JAX imposes some constraints on how these functions should be constructed, it enables complete environment transition functions to operate on the GPU. This allows many more operations and environments to run in parallel and eliminates data transfers between the CPU and GPU for gradient descent updates, both of which can potentially decrease the computational time requirements of reinforcement learning experiments significantly.

4 Environment Design

In many real-world control environments not all state variables are directly affected by the actions of the agent. Instead, some of the state variables transition into their next state via an (agent-independent) function (often time series). These functions often rely on some external data source and therefore these variables describe exactly the entry points for data integration that can be flexibly interchanged within Chargax. Although this data distinction is often implicitly present (Ponse et al., 2024), we will formalise this separation explicitly in Chargax to make clear which parts of the state can flexibly be interchanged.

Consequently, we split the environment state in an *endogenous* and an *exogenous* state space. The endogenous state space refers to the typical state variables that are influenced by the agents' actions during the transition function. In contrast, exogenous state variables transition into their next state via an (agent-independent) time series. Examples of exogenous state variables are weather variables, or national electricity prices. Even though these variables are not affected by the agents' actions, they may influence the agent by providing an additional learning signal and/or alter the reward.

An overview of Chargax is shown in Figure 2 and in the following we provide a high-level overview of the Chargax environment. Full implementation details, including all equations for transition dynamics and reward functions, are provided in Appendix A.

[2] $\Delta(\mathcal{X})$ denotes the set of probability distributions over a set $\mathcal{X}$

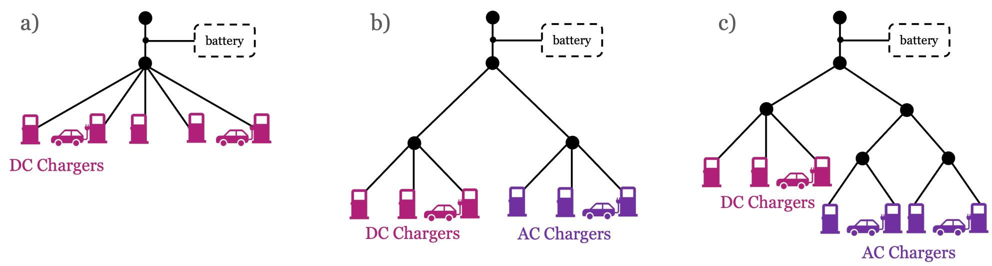


Figure 3: Trees representing different architectures: a) simplest scenario, one type of charger; b) multiple types of chargers, one splitter per charging type; c) multiple types of chargers with multiple splitters per type, imposing additional constraints on the currents. Each node represents a combination of splitters, transformers, cables, and other electrical components.

EV Station Layout

When initialising a Chargax environment, a fixed architectural design for the station is generated or provided. This design is fixed and, therefore, not influenced by the transition function. We represent this electronic infrastructure of the charging station in the form of a tree (Lee et al., 2021), with leaves representing the charging ports (Electric Vehicle Supply Equipment; EVSE; Lee et al. (2020b)) (see Figure 3). The root node represents the grid connection access, and all other nodes represent a combination of splitters, cables, and transformers, and are equipped with a maximum power capacity and efficiency coefficient, imposing constraints on the system. In Chargax, we additionally assume a fixed voltage V for each of the EVSEs in the architecture.

Chargax supplies methods for generating some charging station architectures. However, custom architectures can be built by constructing a tree of simple nodes to mirror existing real-world infrastructure.

Endogenous State Space

The endogenous state consists of the state of the various charging ports and their connected cars, and the station battery. As each charging port (and the battery) has a fixed voltage level, we allow the actual power drawn to be regulated by controlling the current (Orfanoudakis et al., 2024). Losses are incorporated through efficiency coefficients at each node (including the charging ports).

In addition to the set current at each charging port ($I_{\text{drawn}}(t) \in [0, I_{\max}]$), and whether the port is currently occupied ($\mathbb{1}_{\text{occup}}$), the endogenous state contains information for the connected cars. This includes their state-of-charge (SoC) and the remaining required power ΔE_{remain}. Additional information for each car is supplied exogenously and remains fixed until the car leaves. We will expand on this information in the next section. The endogenous state space can optionally be expanded with a station battery. This battery is modelled similarly to an EVSE – with a fixed voltage and controlled via the set current. The battery allows the agent to store energy to facilitate effective discharging strategies. In brief summary, the endogenous state is represented by:

- For each EVSE: $I_{\text{drawn}}(t) \in \mathbb{R}_{\geq 0}$, $\mathbb{1}_{\text{occup}}(t) \in \{0, 1\}$, ΔE_{remain}, SoC(t)
- Battery: $I_{\text{battery}}(t), \text{SoC}_{\text{battery}}(t)$

Enumerating the existing EVSEs by $i = 1, \ldots, N$, the total endogenous state space can be expressed as $s_{\text{end}} = (s_{\text{battery}}, s_{c,1}, \ldots, s_{c,N})$. A complete overview of the state space is given in Appendix A.1.

Exogenous State Space

As described previously, the exogenous state variables evolve independently of the agent's actions. As such, the remainder of the variables discussed here are typically sampled from distributions that are generated via a provided time series or some predefined function. Currently, Chargax works with exogenous state variables for arrival data, user profiles, car profiles, and grid price data.

The **arrival data** represents the number of cars that arrive at a given timestep. Typically, this depends on the time and location of the charging station. Likewise, the location can also stipulate the typical **user profile** of the arriving cars. This profile describes the state of the car that is induced by their owner, such as the arrival SoC, desired charging level, and time of departure. **Car profile** variables are derived from the physical properties of the cars themselves. These include the maximum capacity of the car battery and the maximum charge speed. Lastly, the **grid prices** are an important exogenous variable for calculating the profit, which is often a large factor in the reward.

Chargax comes equipped with a variety of standard datasets (see Table 1), most of which are based on real data. These datasets can be used to sample exogenous variables that resemble realistic scenarios. For example, Europe and the US have a different distribution of electric vehicles on the road; in turn, the distribution of charging demands is different in both regions. While datasets are provided, Chargax is built such that users can use their own data or functions to populate these variables.

Action Space

At each timestep, the agent controlling the charging station can adjust the power at each EVSE by altering the current (Orfanoudakis et al., 2024), i.e. an action is characterized as

$$a(t) = (\Delta I_i(t))_{i=1}^{N+1} \in \mathbb{R}^{N+1}.$$

Here, for the sake of notational convenience, the battery is treated as the $N+1$-th charging pole. Notably, the agent cannot accept/decline cars and is assumed to serve arriving cars, as long as there are free spots.

Transition Function

At a high level, the transition function consists of four sequential steps, which we detail below. Full implementation details can be found in Appendix A.2.

- **Apply Actions** First, we apply the agent's to adjust the power drawn by each charging port. We limit the maximum power by the capacity of the port, as well as the current maximum (dis)charging rate of the car stationed at each charging port.
- **Charge Stationed Cars** With the newly set power levels, we (dis)charge each car over the time interval of a timestep. Here, we assume a constant charging rate over the full interval Δt.
- **Departure of Cars** Next, cars fully charged (charge-sensitive users) or with no time remaining (time-sensitive users) will leave.

Table 1: Overview of available Profiles in Chargax. Default settings are marked in bold.

Price Profiles	Architectures	Car Distributions	Arrival Frequency	User Profiles
NL	Simple: Single Charger Type	**Europe**	Low Traffic	Highway
FR	**Simple: Multiple Charger Types**	US	**Medium Traffic**	Residential
DE	*Custom*	World	High Traffic	Work
Custom		*Custom*	*Custom*	**Shopping**
				Custom

- **Arrival of new Cars** Finally, an amount of new cars will be sampled through our exogenous data, along with a *user profile* and *car profile*. The amount of new cars is clipped by the number of free spots available and the remaining cars are automatically rejected. Arriving cars will park in the first available spot as provided by the provided station architecture.

Reward Function

In RL, the reward functions reflects the notion of optimality, i.e. the desired behaviour. In this section, we outline some of the reward functions that are available in Chargax, and how they reflect different objectives. We provide additional details in Appendix A.3.

Profit Maximisation Profit maximisation lies at the core of most Charging Station Operations (Alinejad et al., 2021; Chang et al., 2021; Mirzaei & Kazemi, 2021; Ye et al., 2022). The amount of net energy transferred into cars in the interval $[t, t + \Delta t]$ is denoted by $\Delta E_{\text{net}}(t)$. The amount of energy fed into the grid as a result from discharging cars is denoted by $\Delta E_{\to\text{grid}}(t)$, and the amount of energy that has to be drawn from the net to transfer set levels of energy into cars $\Delta E_{\text{grid}\to}(t)$. Lastly, the energy contributed by (dis-)charging the battery $\Delta E_{\text{b,net}}(t)$ has to be incorporated, resulting in the following net energy that is drawn from (or pushed into) the grid

$$\Delta E_{\text{grid,net}} = \Delta E_{\text{grid}\to}(t) + \Delta E_{\to\text{grid}}(t) + \Delta E_{\text{b,net}}(t). \tag{1}$$

We further assume that the price at which we sell and buy power from car owners is the same, i.e. p_{sell}. This results in the following profit

$$\Pi(t) = \begin{cases} p_{\text{sell}}(t) \cdot \Delta E_{\text{net}}(t) - p_{\text{buy}}(t) \cdot \Delta E_{\text{grid,net}} - c_{\Delta t} & \Delta E_{\text{grid,net}} > 0, \\ p_{\text{sell}}(t) \cdot \Delta E_{\text{net}}(t) - p_{\text{sell,grid}}(t) \cdot \Delta E_{\text{grid,net}} - c_{\Delta t} & \Delta E_{\text{grid,net}} \leq 0. \end{cases} \tag{2}$$

Here, $c_{\Delta t}$ denotes the fixed cost for running the facility per Δt.

Profit Maximisation under constraints To further steer agents' learnt behaviour in a direction, constraints can be induced to penalise certain (undesired) behaviour through penalty terms $c(t)$. The resulting reward will be the profit minus the linear combination of (possibly) multiple penalty terms

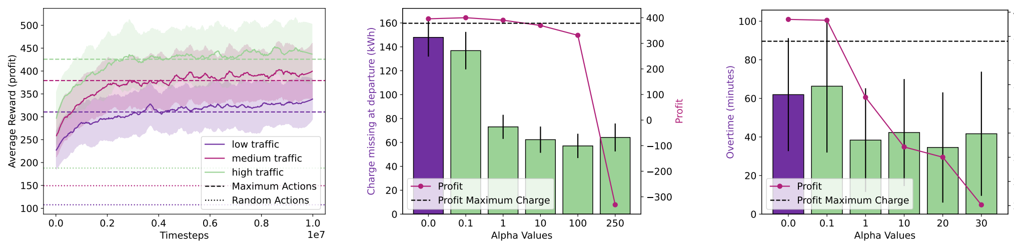
$$r(t) = \Pi(t) - \sum_c \alpha_c \, c(t). \tag{3}$$

Different linear combinations of different penalty terms allow Chargax to be flexible in its optimization objective. Chargax comes equipped with various of these penalty terms to better optimize for, for example, customer satisfaction, battery degradation, or violating node constraints. We provide a more complete list of possible penalty terms along with a formal expression in Appendix A.3. However, we emphasise that these are mere suggestions, and that these rewards are not comprehensive in reflecting the full landscape of Charging Station Optimisation challenges, and we encourage users to customise their reward function within the provided framework.

5 Experiments

In this section, we demonstrate the use of Chargax across different included scenarios. Additionally, we highlight performance improvements of Chargax compared to previous EV charging simulations. Full details of the used model and configuration parameters, along with additional experimental results, can be found in Appendix B and D respectively.

In Figure 4a, we have trained a standard PPO agent based on PureJaxRL (Lu et al., 2022). We trained on our included *shopping* scenario in varying amounts of traffic using a 16 charger station (10 DC, 6 AC). We observe how our PPO agent increases its profit over a standard baseline. The baseline is set to always charge to its maximum potential within the constraints of the EVSE and the



(a) PPO in Shopping scenario (b) Charge missing at departure (c) Charged overtime

Figure 4: In a) average episode rewards during training a PPO agent in the shopping scenario with different levels of traffic. The RL solution manages to increase profit over the baseline that always charged the maximum possible amount. In b) and c), user satisfaction measured as charge (kWh) missing at time of departure (b), and time exceeded to fully charge cars (c). Higher α-values weigh the measured variable greater in the reward (Eq. 3). Increasing user satisfaction tends to decrease daily profit. Notably however in b), optimizing for user satisfaction has steered the agent to find policies that reduce the missing charge percentages while retaining a near-identical profit level.

connected car. As expected, the potential for profit increases in scenarios with higher amounts of traffic, but this increase diminishes as we kept the charging station size the same.

Our baseline should yield a high customer satisfaction as customers should be charged within the minimum amount of possible time. In contrast, our charging station agent may optimize fully for short-term profit without consideration of user satisfaction. This is likely undesirable and may affect long-term profits. However, Chargax allows for flexible reward signals that may optimize for this. In Figure 4b and 4c, we trained our PPO agent to optimize for profit and user satisfaction at varying α levels. Notably in Figure 4b, we can see the agent manages to find preferential policies that substantially increase user satisfaction (decrease the amount of kWh that was not charged at departure time), while keeping profit levels quite similar.

Beyond finding appropriate reward signals, real-world deployment typically involves training an agent on historical exogenous data. During deployment, the agent likely encounters data that is has not yet observed. Possibly, the entire data set has shifted, for example, due to a rise in energy prices year-over-year. Therefore, it is important that system that deal with exogenous time-series data can deal with – and test for – this distribution shift (Yeh et al., 2024). As Chargax is flexibly designed to allow for any exogenous data, it readily allows to test for these distribution shift problems – as is displayed in Figure 5, where we have trained and evaluated RL agents on data of different

Table 2: Performance comparison between Chargax and other EV charging Gym environments, based on data collected by performing 100k environment steps. We evaluated both taking random actions (assesing the performance of the transition function), and a training a PPO agent. The PPO agent was tested both with a single environment, and in a more typical training scenario with vectorized environments. Here we observe performance improvements of over 100x. The results are obtained on an `NVIDIA RTX 4000 Ada GPU` and an `AMD EPYC 2.8 GHz CPU`. For the comparison environments, we used Stable-Baselines3 (Raffin et al., 2021) with CUDA enabled for the PPO implementation.

	Chargax	Ev2Gym		Chargym		Sustaingym	
			Speedup		**Speedup**		**Speedup**
Random	1.36	77.95	57x	36.34	27x	1554.57	1144x
PPO (1)	9.79	170.05	17x	131.18	13x	1718.71	176x
PPO (16)	0.65	86.99	**134x**	125.06	**192x**	1836.00	**2820x**

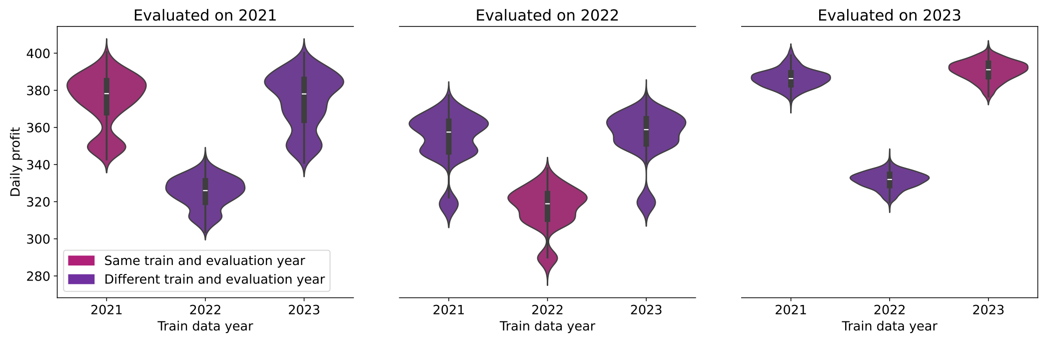


Figure 5: A PPO agent trained trained and tested on three seperate years of Dutch electricity prices. For each experiment, a particular year's data was used for training, while testing on a fixed year. Substantial price increases in the year 2022 results in suboptimal training when using this year's data – even when evaluating on this same year.

price electricity years. Interestingly, although rewards would be assumed to peak when training and testing in the same year, employing data from 2021 or 2023 actually yielded higher rewards in 2022 compared to using the 2022 data directly. The EU region experienced significant energy price surges in 2022, likely complicating the training process with the data for this year.

Table 2 and Figure 1, showcases the performance of our environment compared to existing EV charging simulations that support reinforcement learning through a Gym API. We can see that in a typical training scenario, we can decrease learning times by factors exceeding 100. It is important to acknowledge that these environments are not identical and might simulate different behaviours (for example, SustainGym does not allow discharging). Therefore, this comparison may be considered rough. However, the significant differences in scale clearly demonstrate the advantages of using Chargax- and JAX-based environments for RL in general. Training cycles can be reduced entire working days to well under 5 minutes, allowing for many more iterations of training and testing.

6 Discussion & Conclusion

This work presented Chargax, an EV charging simulator built in JAX. Chargax aims to bridge the gap between toy problems and real-world implementations, accelerating simulations while maintaining practical relevance. However, it remains a simulator, constrained by simplifying assumptions, requiring future work to further close the gap between simulation and deployment.

Our model assumes an isolated power network for the EV charging station, avoiding shared transformers that could introduce uncontrollable constraints. Expanding the model to include additional control variables, such as dynamic pricing strategies or vehicle allocation mechanisms, would increase its realism. Furthermore, accounting for temperature dependence in the system, or incorporating government-imposed regulatory constraints could make it more reflective of real-world charging stations. Furthermore, a natural addition for future work would be to incorporate local energy production systems (such as solar panels) and weather data.

In its current state, Chargax achieves training time reductions of over 100x, compared to existing simulators. Usual training durations of (multiple) working days can be completed in Chargax in well under 5 minutes, allowing for many more additional training and testing runs. We have built Chargax to be flexible, allowing for custom data sources for the exogenous state, and flexible reward structures. However, Chargax does provide base datasets and reward penalties to get started. Chargax can also be used as a test bed reinforcement learning benchmarking as it is currently only one of few JAX-based environments that models a real-world problem. We open source Chargax for the wider community to experiment with.[1]

A Environmental Details

A.1 State Spaces

Exogenous state space Apart from price data, examples of exogenous state variables include power demand of the grid, weather data, or marginal operating emissions rate (MOER, Yeh et al., 2024), all of which could influence the maximisation objective but evolve according to some (agent independent) time series. It is important to note, that while the environment requires auxiliary data for most built-in reward functions, e.g. it is impossible to maximise profit without having access to prices, these exogenous state variables may be treated unobservable for the agent. On the contrary, one may add data to the exogenous state space, that is not required for any reward calculation, but may serve as additional learning signal, for instance day-ahead power prices.

Apart from the above examples arrival data, user profiles, and car profiles are part of the exogenous state space.

- **Arrival Data** At each timestep t, a number of cars $M(t)$ is characterized as a sample from an arrival distribution $M(t) \sim \mathcal{D}_{\text{arrival}}(t)$.
- **Car Profiles** Arriving cars are characterised by their physical properties. This encompasses the charging speed $\hat{r}$ as a function of the SoC. As in (Lee et al., 2020b) we assume a piece-wise linear function
$$\hat{r}_{\tau,\bar{r}}(\text{SoC}) = \begin{cases} \bar{r}, & \text{SoC} \leq \tau \\ (1 - \text{SoC})\frac{\bar{r}}{1-\tau}, & \text{SoC} > \tau. \end{cases}$$
Due to lack of data, we assume that the discharging speed can be obtained by vertically flipping the charging curve at $\text{SoC} = 0.5$. While we assume, that we have a different maximal charging speed for different charger types – by default AC and DC charger – and have hence different max charging rates ($\bar{r} = (\bar{r}_{\text{AC}}, \bar{r}_{\text{DC}})$), we assume that both charging speed curves use the same τ. Lastly, each car has a maximum battery capacity C, which is important for calculating State of Charges. These car profiles are sampled from a pre-defined car distribution $\mathcal{D}_{\text{car}}(t)$, see also Table 1.
- **User Profiles** Additionally to the physical properties, the charging demand is a result from the habits of the car owner, encompassing a duration of stay Δt_{remain}, the number of units of power to be charged ΔE, the SoC upon arrival SoC_0 and the user preference u, indicating whether a user is time-sensitive (will leave iff $\Delta t_{\text{remain}} = 0$), or charge sensitive (will leave iff $\Delta E_{\text{remain}} = 0$). The user profiles are sampled from a distribution $\mathcal{D}_{\text{user}}(t)$, see also Table 1.

Endogenous state space The endogenous state consists of the state of the various charging ports and their connected cars, and the station battery. For each charging port, we assume a fixed voltage and allow the actual power drawn to be regulated by controlling the current $I_{\text{drawn}}(t) \in [0, I_{\max}]$ (Orfanoudakis et al., 2024). We assume that the voltage value already encodes the phases, i.e. it represents the product $V \cdot \sqrt{\phi}$ in Orfanoudakis et al. (2024), eliminating the need for the phase as an additional variable. To incorporate losses during the (dis)charging process, each EVSE is equipped with an efficiency coefficient for charging and discharging. As a charging port may not always be occupied, we add a final Boolean to the state $\mathbb{1}_{\text{occup}}$, indicating the presence of a car.

To properly facilitate discharging, the charging station is equipped with a battery. Similarly to EVSEs, the battery will have a fixed voltage V_{battery}, with the power flow controlled by the current $I_{\text{battery}}(t)$. To specify the physical properties of the battery, it also has a maximum capacity C, the maximal charging rate for a car $\bar{r}$ and $\tau \in (0, 1)$. Additionally, we will equip the state with the current SoC of the battery

$$s_{\text{battery}} = (I_{\text{battery}}(t), \text{SoC}_{\text{battery}}(t), \hat{r}_{\text{battery}}(t)).$$

Car State Additionally, the state of each charging port contains information for the connected cars, the so-called car state, representing the car that is charging at this port (all zeros if no car is present).

As this car state consists of exogenous and endogenous variables, it is listed separately. This includes the car's state-of-charge (SoC $\in [0,1]$), the remaining required power $\Delta E_{\text{remain}} \in \mathbb{R}_{\geq 0}$, the number of timesteps the car remains $\Delta t_{\text{remain}} \in \mathbb{N}$, and the maximal charging power currently allowed by the car $\hat{r}(t) \in \mathbb{R}_{\geq 0}$. The latter one is heavily depended on the State of Charge $\text{SoC}(t) \in [0,1]$ of the car battery (Welzel et al., 2021; Fastned, 2025), which is also part of the car-state. The car-state also contains information about the physical properties of the car. These are the maximum battery capacity C, the maximum charging rate for a car $\bar{r}$, and $\tau \in (0,1)$ – the transition point from the bulk stage to the absorption stage of the charging process (Lee et al., 2020b). Finally, the car-state includes a user preference indicator u.

In brief summary, the state of each charging port is represented by:

- Current power drawn $I_{\text{drawn}}(t) \in \mathbb{R}_{\geq 0}$, occupancy indicator $\mathbb{1}_{\text{occup}}(t) \in \{0,1\}$;
- Car-state $(\Delta E_{\text{remain}}(t), \Delta t_{\text{remain}}(t), \hat{r}(t), \text{SoC}(t), C, \bar{r}, \tau, u)$.

A.2 Transition Function

The transition function consists of four major steps: (i) Apply Actions, i.e. adapt charging levels at each EVSE, (ii) charge stationed cars, (iii) departure of cars, and (iv) arrival of new cars.

Apply Actions As a first step, the actions taken by the agent are applied to adjust the power drawn by each charging pole, specifically

$$I_{\text{drawn},i}(t) = \begin{cases} \min\left(I_{\text{drawn},i}(t-\Delta t) + a_i(t), \hat{r}(t), I_{\max\to,\text{i}}\right) & I_{\text{drawn},i}(t-\Delta t) + a_i(t) \geq 0 \\ -\min\left(-I_{\text{drawn},i}(t-\Delta t) - a_i(t), \hat{r}(t), I_{\max\leftarrow,\text{i}}\right) & \text{else.} \end{cases}$$

Hereby constraints on the maximum power drawn imposed by the architecture are enforced by assuring that for each subtree H in the architecture, the constraints

$$\frac{1}{\eta_H} \sum_{h \in \text{leaves}(H)} I_{\text{drawn},h}(t) \leq I_H, \tag{4}$$

are satisfied. If the drawn currents violate these constraints, the currents at each leaf are rescaled to satisfy the constraints, modelling the potential behaviour of some safety infrastructure on top of the controller.

Charge Stationed Cars After having adjusted the power levels at each charging pole, the charging is processed for the time interval, where a constant charging rate over the full interval Δt is assumed. The car states are adjusted in the following way:

$$\begin{aligned} \Delta E_{\text{remain},i}(t+\Delta t) &= \Delta E_{\text{remain},i}(t) - \Delta t \cdot V_i \cdot I_{\text{drawn},i}(t) \\ \text{SoC}(t+\Delta t) &= \text{SoC}(t) + \frac{\Delta t \cdot V_i \cdot I_{\text{drawn},i}(t)}{C_i} \\ \hat{r}(t+\Delta t) &= \hat{r}_{\tau_i,\bar{r}_i}(\text{SoC}(t+\Delta t)). \end{aligned}$$

Notably, the physical attributes of the car in the car state, i.e. the maximum battery capacity, the maximal charging rate and τ do not change. As charging has been proceed, we assume that time moves on, i.e. $t \mapsto t + \Delta t$ and $\Delta t_{\text{remain},i}(t+\Delta t) = \Delta t_{\text{remain},i}(t) - \Delta t$.

Departure of Cars At the end of the period, cars fully charged or with no time remaining will leave. Consequently the car-states for the corresponding charging poles are updated

$$s_{c,i}(t) = \begin{cases} (0,\ldots,0) & \Delta t_{\text{remain},i}(t) = 0 \text{ and } u_i = 0 \\ (0,\ldots,0) & \Delta E_{\text{remain},i}(t) = 0 \text{ and } u_i = 1 \\ s_{c,i}(t) & \text{else.} \end{cases}$$

Arrival of new Cars The amount of arriving cars is sampled $M(t) \sim \mathcal{D}_{\text{arrival}}(t)$. We model a first-come-first-served policy by clipping $M(t)$ by the number of available free spots $N - \sum_{i=1}^{N} \mathbb{1}_{\text{occup},i}(t)$. For each car $j = 1, \dots, M(t)$ the car profile, and the user profile are sampled from their respective distribution, i.e. $(\Delta t_{\text{remain},j},\, \Delta E_j,\, \text{SoC}_{0,j}, u_j) \sim \mathcal{D}_{\text{profile}}(t)$ and $(\bar{r}_j, \tau_j, C_j) \sim \mathcal{D}_{\text{car}}(t)$, respectively.

Each car j is then allocated to a free charging pole k, which alters the state of charging pole k based on car j:

$$s_{c,k}(t) = (0, 1, \Delta E_j, \Delta t_{\text{remain},j}, \hat{r}_{\tau_j,\bar{r}_j}(\text{SoC}_{0,j}), C_j, \bar{r}_j, \tau_j, u_j).$$

A.3 Reward functions

The amount of net energy transferred into cars in the interval $[t, t + \Delta t]$ can be calculated as $\Delta E_{\text{net}}(t) = \Delta t \sum_{i=1}^{N} V_i \cdot I_{\text{drawn},i}(t)$. Accounting for losses within the electric architecture of the charging station, the amount of energy, that is transferred from cars into the grid can be calculated as

$$\Delta E_{\rightarrow\text{grid}}(t) = \Delta t \sum_{i: I_{\text{drawn},i} < 0} \eta_i \cdot V_i \cdot I_{\text{drawn},i}(t) < 0. \tag{5}$$

Similarly, the amount of energy that has to be drawn from the net to transfer set levels of energy into cars $\Delta E_{\text{grid}\rightarrow}(t)$, after incorporating imperfect efficiencies, can be calculated via $\Delta E_{\text{grid}\rightarrow}(t) = \Delta t \sum_{i: I_{\text{drawn},i} > 0} \eta_i^{-1} \cdot V_i \cdot I_{\text{drawn},i}(t) > 0$. Lastly, the energy contributed by (dis-)charging the battery $\Delta E_{\text{b,net}}(t) = \Delta t\, I_{\text{battery}}(t)\, V_{\text{battery}}$ has to be incorporated, resulting in the following net energy drawn from (or pushed into) the grid

$$\Delta E_{\text{grid,net}} = \Delta E_{\text{grid}\rightarrow}(t) + \Delta E_{\rightarrow\text{grid}}(t) + \Delta E_{\text{b,net}}(t).$$

Further that the price at which we sell and buy power from car owners is the same, i.e. p_{sell}. This results in the following revenue

$$\Pi(t) = \begin{cases} p_{\text{sell}}(t) \cdot \Delta E_{\text{net}}(t) - p_{\text{buy}}(t) \cdot \Delta E_{\text{grid,net}} - c_{\Delta t} & \Delta E_{\text{grid,net}} > 0, \\ p_{\text{sell}}(t) \cdot \Delta E_{\text{net}}(t) - p_{\text{sell,grid}}(t) \cdot \Delta E_{\text{grid,net}} - c_{\Delta t} & \Delta E_{\text{grid,net}} \leq 0. \end{cases}$$

Here, $c_{\Delta t}$ denotes the fixed cost for running the facility per Δt. The general reward $r(s(t), a(t), s(t + \Delta t))$, abbreviated by $r(t)$ in Chargax consists of the profit minus the linear combination of some penalty terms

$$r(t) = \Pi(t) - \sum_{c} \alpha_c c(t). \tag{6}$$

Some examples of included penalty terms are listed below

- **Constraint Violations** The hard constraints imposed by the architecture in Eq. 4 could be instead included as as soft constraints (Yeh et al., 2024) via the penalty

$$c_{\text{constraint}}(t) = \max_{H} \min \left(0, \frac{1}{\eta_H} \sum_{i \in \text{leaves}(H)} I_{\text{drawn},i}(t) - I_H \right).$$

- **Satisfaction penalty** Users can experience dissatisfaction in two ways: Time-sensitive users have a desired departure time and are assumed to leave at that time, regardless the SoC of their car. To avoid customers leaving the charging station with a suboptimal SoC we propose to incorporate a satisfaction penalty

$$c_{\text{Satisfcation},0}(t) = \sum_{i: \Delta t_{\text{remain},i}(t) = 0, u_i = 0} \max(0, \Delta E_{\text{remain,i}}(t)).$$

The opposite holds for charge sensitive users, as they are expected to leave when there cars are charged to the desired level. However, these users can be overly satisfied by charging their car to the desired level faster than desired

$$c_{\text{Satisfcation},1}(t) = \sum_{i:\Delta E_i(t)=0,u_i=1} \max(0, -\Delta t_{\text{remain},i}(t)) - \beta \max(0, \Delta t_{\text{remain},i}(t)).$$

Here β controls how much the positive satisfaction from leaving earlier should weight in comparison to the negative dissatisfaction from having to stay overtime.

- **Sustainability** To enforce the agent to charge cars in the most sustainable way possible, a penalty term for non-sustainable behaviour may be added. One solution proposed in (Yeh et al., 2024) is to employ the MOER $m(t)$, capturing the carbon intensity of a unit of energy produced at time t

$$c_{\text{sustain}}(t) = m(t) \cdot \Delta E_{\text{grid,net}}(t).$$

- **Rejected Customers** In view of congestion management problems (Zhang et al., 2019; Hussain et al., 2022), one might be interested in serving the maximum number of cars, i.e. reduce the amount of rejected cars, by adding a penalty term for declined cars

$$c_{\text{declined}}(t) = \max\left(M(t) - \left(N - \sum_{i=1}^{N} \mathbb{1}_{\text{occup},i}(t)\right), 0\right).$$

- **Battery Degradation** Real world batteries suffer from degradation under use (Lee et al., 2020a). This can be incorporated by adding a degradation cost to every discharging of the Charging station battery, as well as for the cars. For sake of simplicity, we assume the additional degradation to be proportional to the discharged energy

$$c_{\text{degrad,battery}}(t) = |\Delta E_{\text{b,net}}(t)| \cdot \mathbb{1}_{\{\Delta E_{\text{b,net}}(t)<0\}} \text{ and } c_{\text{degrad,cars}}(t) = |\Delta E_{\rightarrow\text{grid}}(t)|.$$

- **Grid Stability** (Only applicable in a V2G scenario) If the agent can discharge cars, this can be leveraged to stabilize the grid load (Li et al., 2021; Elma, 2020). This could be reflected in a penalty term through an exogenous signal of the grid demand $d_{\text{grid}}(t) \in \mathbb{R}$

$$c_{\text{grid}}(t) = |\Delta E_{\text{net}}(t) - d_{\text{grid}}(t)|.$$

References

Pegah Alaee, Julius Bems, and Amjad Anvari-Moghaddam. A Review of the Latest Trends in Technical and Economic Aspects of EV Charging Management. *Energies*, 16(9):3669, January 2023. ISSN 1996-1073. DOI: 10.3390/en16093669.

Mahyar Alinejad, Omid Rezaei, Ahad Kazemi, and Saeed Bagheri. An Optimal Management for Charging and Discharging of Electric Vehicles in an Intelligent Parking Lot Considering Vehicle Owner's Random Behaviors. *Journal of Energy Storage*, 35:102245, March 2021. ISSN 2352152X. DOI: 10.1016/j.est.2021.102245.

Emmanuel Balogun, Elizabeth Buechler, Siddharth Bhela, Simona Onori, and Ram Rajagopal. Evecosim: A grid-aware co-simulation platform for the design and optimization of electric vehicle charging infrastructure. *IEEE Transactions on Smart Grid*, 2023.

Clément Bonnet, Daniel Luo, Donal Byrne, Shikha Surana, Sasha Abramowitz, Paul Duckworth, Vincent Coyette, Laurence I. Midgley, Elshadai Tegegn, Tristan Kalloniatis, Omayma Mahjoub, Matthew Macfarlane, Andries P. Smit, Nathan Grinsztajn, Raphael Boige, Cemlyn N. Waters, Mohamed A. Mimouni, Ulrich A. Mbou Sob, Ruan de Kock, Siddarth Singh, Daniel Furelos-Blanco, Victor Le, Arnu Pretorius, and Alexandre Laterre. Jumanji: a diverse suite of scalable reinforcement learning environments in jax, 2024. URL https://arxiv.org/abs/2306.09884.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/jax-ml/jax.

M. Cañigueral. evsim: Electric vehicle charging sessions simulation, 2023. R package version 1.2.0. [Online]. Available: https://github.com/mcanigueral/evsim/.

Shuo Chang, Yugang Niu, and Tinggang Jia. Coordinate scheduling of electric vehicles in charging stations supported by microgrids. *Electric Power Systems Research*, 199:107418, October 2021. ISSN 03787796. DOI: 10.1016/j.epsr.2021.107418.

Onur Elma. A dynamic charging strategy with hybrid fast charging station for electric vehicles. *Energy*, 202:117680, July 2020. ISSN 03605442. DOI: 10.1016/j.energy.2020.117680.

Fastned. Charge speed, 2025. URL https://www.fastnedcharging.com/en/brands-overview. Accessed: 2025-02-14.

Borja Fernandez-Gauna, Manuel Graña, Juan-Luis Osa-Amilibia, and Xabier Larrucea. Actor-critic continuous state reinforcement learning for wind-turbine control robust optimization. *Information Sciences*, 591:365–380, April 2022. ISSN 00200255. DOI: 10.1016/j.ins.2022.01.047.

C. Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. Brax - a differentiable physics engine for large scale rigid body simulation, 2021. URL http://github.com/google/brax.

Jonathan Heek, Anselm Levskaya, Avital Oliver, Marvin Ritter, Bertrand Rondepierre, Andreas Steiner, and Marc van Zee. Flax: A neural network library and ecosystem for JAX, 2024. URL http://github.com/google/flax.

Shahid Hussain, Yun-Su Kim, Subhasis Thakur, and John G. Breslin. Optimization of Waiting Time for Electric Vehicles Using a Fuzzy Inference System. *IEEE Transactions on Intelligent Transportation Systems*, 23(9):15396–15407, September 2022. ISSN 1558-0016. DOI: 10.1109/TITS.2022.3140461.

Lukasz Kaiser, Mohammad Babaeizadeh, Piotr Milos, Blazej Osinski, Roy H. Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, Afroz Mohiuddin, Ryan Sepassi, George Tucker, and Henryk Michalewski. Model-Based Reinforcement Learning for Atari, April 2024. URL http://arxiv.org/abs/1903.00374. arXiv:1903.00374 [cs, stat].

Georgios Karatzinis, Christos Korkas, Michalis Terzopoulos, Christos Tsaknakis, Aliki Stefanopoulou, Iakovos Michailidis, and Elias Kosmatopoulos. Chargym: An ev charging station model for controller benchmarking. In *IFIP International Conference on Artificial Intelligence Applications and Innovations*, pp. 241–252. Springer, 2022.

Patrick Kidger and Cristian Garcia. Equinox: neural networks in JAX via callable PyTrees and filtered transformations. *Differentiable Programming workshop at Neural Information Processing Systems 2021*, 2021.

Sotetsu Koyamada, Shinri Okano, Soichiro Nishimori, Yu Murata, Keigo Habara, Haruka Kita, and Shin Ishii. Pgx: Hardware-accelerated parallel game simulators for reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 45716–45743, 2023.

Robert Tjarko Lange. gymnax: A JAX-based reinforcement learning environment library, 2022. URL http://github.com/RobertTLange/gymnax.

Munsu Lee, Jinhyeong Park, Sun-Ik Na, Hyung Sik Choi, Byeong-Sik Bu, and Jonghoon Kim. An Analysis of Battery Degradation in the Integrated Energy Storage System with Solar Photovoltaic Generation. *Electronics*, 9(4):701, April 2020a. ISSN 2079-9292. DOI: 10.3390/electronics9040701.

Zachary J. Lee, Sunash Sharma, Daniel Johansson, and Steven H. Low. ACN-Sim: An Open-Source Simulator for Data-Driven Electric Vehicle Charging Research. https://arxiv.org/abs/2012.02809v2, December 2020b.

Zachary J. Lee, George Lee, Ted Lee, Cheng Jin, Rand Lee, Zhi Low, Daniel Chang, Christine Ortega, and Steven H. Low. Adaptive Charging Networks: A Framework for Smart Electric Vehicle Charging. *IEEE Transactions on Smart Grid*, 12(5):4339–4350, September 2021. ISSN 1949-3061. DOI: 10.1109/TSG.2021.3074437.

Yang Li, Meng Han, Zhen Yang, and Guoqing Li. Coordinating Flexible Demand Response and Renewable Uncertainties for Scheduling of Community Integrated Energy Systems With an Electric Vehicle Charging Station: A Bi-Level Approach. *IEEE Transactions on Sustainable Energy*, 12 (4):2321–2331, October 2021. ISSN 1949-3037. DOI: 10.1109/TSTE.2021.3090463.

Chris Lu, Jakub Kuba, Alistair Letcher, Luke Metz, Christian Schroeder de Witt, and Jakob Foerster. Discovered policy optimisation. *Advances in Neural Information Processing Systems*, 35:16455–16468, 2022.

Mohammad Javad Mirzaei and Ahad Kazemi. A two-step approach to optimal management of electric vehicle parking lots. *Sustainable Energy Technologies and Assessments*, 46:101258, August 2021. ISSN 22131388. DOI: 10.1016/j.seta.2021.101258.

Alexander Nikulin, Vladislav Kurenkov, Ilya Zisman, Viacheslav Sinii, Artem Agarkov, and Sergey Kolesnikov. XLand-minigrid: Scalable meta-reinforcement learning environments in JAX. In *Intrinsically-Motivated and Open-Ended Learning Workshop, NeurIPS2023*, 2023. URL https://openreview.net/forum?id=xALDC4aHGz.

Stavros Orfanoudakis, Cesar Diaz-Londono, Yunus E. Yılmaz, Peter Palensky, and Pedro P. Vergara. EV2Gym: A Flexible V2G Simulator for EV Smart Charging Research and Benchmarking, April 2024.

Eduardo Pignatelli, Jarek Liesen, Robert Tjarko Lange, Chris Lu, Pablo Samuel Castro, and Laura Toni. Navix: Scaling minigrid environments with jax. *arXiv preprint arXiv:2407.19396*, 2024.

Koen Ponse, Felix Kleuker, Márton Fejér, Álvaro Serra-Gómez, Aske Plaat, and Thomas Moerland. Reinforcement learning for sustainable energy: A survey. *arXiv preprint arXiv:2407.18597*, 2024.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-Baselines3: Reliable Reinforcement Learning Implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021. ISSN 1533-7928. URL http://jmlr.org/papers/v22/20-1364.html.

Anis ur Rehman, Haris M Khalid, and SM Muyeen. Grid-integrated solutions for sustainable ev charging: a comparative study of renewable energy and battery storage systems. *Frontiers in Energy Research*, 12:1403883, 2024.

Emmanouil S Rigas, Sotiris Karapostolakis, Nick Bassiliades, and Sarvapali D Ramchurn. Evlibsim: A tool for the simulation of electric vehicles' charging stations using the evlib library. *Simulation Modelling Practice and Theory*, 87:99–119, 2018.

Alexander Rutherford, Benjamin Ellis, Matteo Gallici, Jonathan Cook, Andrei Lupu, Gardar Ingvarsson, Timon Willi, Akbir Khan, Christian Schroeder de Witt, Alexandra Souly, Saptarashmi Bandyopadhyay, Mikayel Samvelyan, Minqi Jiang, Robert Tjarko Lange, Shimon Whiteson, Bruno Lacerda, Nick Hawes, Tim Rocktaschel, Chris Lu, and Jakob Nicolaus Foerster. Jaxmarl: Multi-agent rl environments in jax. *arXiv preprint arXiv:2311.10090*, 2023.

S. Saxena. Vehicle-to-grid simulator, version 00, November 2013. [Online]. Available: https://www.osti.gov/biblio/1437011.

Tim Scarfe, Jakob Foerster, and Chris Lu. Machine learning street talk - imagenet moment for reinforcement learning?, 2 2025.

Jaemin Seo, SangKyeun Kim, Azarakhsh Jalalvand, Rory Conlin, Andrew Rothstein, Joseph Abbate, Keith Erickson, Josiah Wai, Ricardo Shousha, and Egemen Kolemen. Avoiding fusion plasma tearing instability with deep reinforcement learning. *Nature*, 626(8000):746–751, 2024.

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.

Fynn Welzel, Carl-Friedrich Klinck, Yannick Pohlmann, and Mats Bednarczyk. Grid and user-optimized planning of charging processes of an electric vehicle fleet using a quantitative optimization model. *Applied Energy*, 290:116717, May 2021. ISSN 03062619. DOI: 10.1016/j.apenergy.2021.116717.

Denis Yarats, Amy Zhang, Ilya Kostrikov, Brandon Amos, Joelle Pineau, and Rob Fergus. Improving Sample Efficiency in Model-Free Reinforcement Learning from Images, July 2020. URL http://arxiv.org/abs/1910.01741. arXiv:1910.01741 [cs].

Zuzhao Ye, Yuanqi Gao, and Nanpeng Yu. Learning to Operate an Electric Vehicle Charging Station Considering Vehicle-Grid Integration. *IEEE Transactions on Smart Grid*, 13(4):3038–3048, July 2022. ISSN 1949-3061. DOI: 10.1109/TSG.2022.3165479.

Christopher Yeh, Victor Li, Rajeev Datta, Julio Arroyo, Nicolas Christianson, Chi Zhang, Yize Chen, Mohammad Mehdi Hosseini, Azarang Golmohammadi, Yuanyuan Shi, et al. Sustaingym: Reinforcement learning environments for sustainable energy systems. *Advances in Neural Information Processing Systems*, 36, 2024.

Yongmin Zhang, Pengcheng You, and Lin Cai. Optimal Charging Scheduling by Pricing for EV Charging Station With Dual Charging Modes. *IEEE Transactions on Intelligent Transportation Systems*, 20(9):3386–3396, September 2019. ISSN 1558-0016. DOI: 10.1109/TITS.2018.2876287.

Supplementary Materials

The following content was not necessarily subject to peer review.

B Implementation Details

B.1 Practical Considerations

Table 3 contains environment settings used throughout our experiments whenever not stated. Additionally, we list some practical considerations in Chargax here.

- The episode length defaults to the length of data provided for arriving cars. In our bundled scenarios, this equals 24 hours. These bundled scenarios provide their data as average numbers per timestep. The actual number of cars arriving is then drawn using a Poisson distribution.
- By default, we train in a Chargax environment utilizing a method akin to exploring starts. At environment reset, we sample a random day from the given price data and use this day's prices for the episode. The agent observes the current episode day and whether this is a weekday or a workday.
- Throughout our experiments, we have used a discretised action space, setting the (user-defined) discretization level to 10. This allows the agent to select increments as 10%, 20%, 30%, etc., up to 100% of the maximum current for each charging port.

B.2 Agent configuration

Unless otherwise stated, the experiments conducted in Section 5 and Appendix D trained with a PPO agent using the hyperparameters listed in Table 3.

Hyperparameter	Value	Environment Parameter	Value
Total timesteps	1e7	Minutes per timestep Δt	5
Learning rate (α)	2.5e-4 (annealed)	Discretization factor	10
Discount factor γ	0.99	Episode length	24 hours
GAE λ	0.95	Number of Chargers	16
Max grad norm	100.0	Number of DC Chargers	10
Clipping coefficient ϵ	0.2	Sell price to customers (p_{sell})	0.75
Value func clip coefficient	10.0	All reward coefficients α (Eq. 3)	0.0
Entropy coefficient	0.01		
Value function coefficient	0.25		
Vectorized environments	12		
Rollout length (steps)	300		
Number of minibatches	4		
Update epochs	4		
Minibatch size	900		
Batch size	3600		

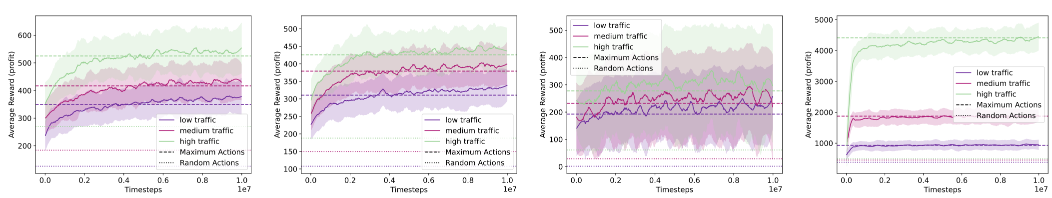
Table 3: PPO hyperparameters (left) alongside environment settings (right) used throughout our experiments unless otherwise stated.

C State summary

Table 4: Summary of the state space in Chargax

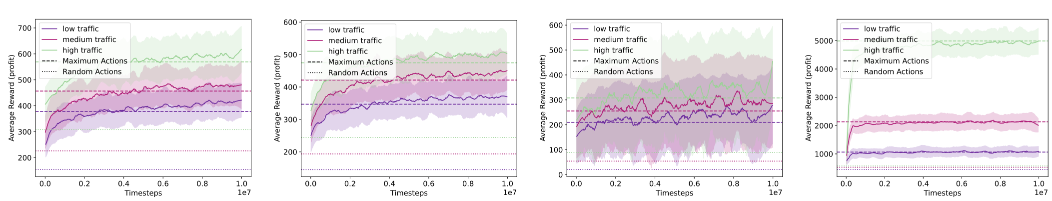
	symbol	domain	exogenous/ endogeneous	variable name
reward data	p_{sell}	$\mathbb{R}_{\geq 0}$	exogenous	Selling price (to Customer) per kWh
	p_{buy}	$\mathbb{R}_{\geq 0}$	exogenous	Buying price per kWh
	$p_{\text{sell,grid}}$	$\mathbb{R}_{\geq 0}$	exogenous	Selling price (to grid) per kWh
	m	$\mathbb{R}_{\geq 0}$	exogenous	Marginal Operations Emission Rate
	d_{grid}	$\mathbb{R}$	exogenous	Grid Demand
	M	$\mathbb{N}_0$	exogenous	Number of arriving cars
Car state of EVSE i	$\Delta t_{\text{remain,i}}$	$\mathbb{N}_0$	exogenous	Remaining time of customer
	C_i	$\mathbb{R}_{\geq 0}$	exogenous	Capacity of Car
	$\bar{r}_i$	$\mathbb{R}_{\geq 0}$	exogenous	Maximum charging rate
	$\hat{r}_i$	$\mathbb{R}_{\geq 0}$	exogenous	Maximum charging rate at current SoC
	τ_i	$[0,1]$	exogenous	
	u_i	$\{0,1\}$	exogenous	User preference
	SoC_i	$[0,1]$	endogenous	Current SoC
	$\Delta E_{\text{remain,i}}$	$\mathbb{R}_{\geq 0}$	endogenous	Remaining Charging demand
State variables of EVSE i	$\mathbb{1}_{\text{occup,i}}$	$\{0,1\}$	endogenous	Occupancy Indicator
	$I_{\text{drawn,i}}$	$\mathbb{R}_{\geq 0}$	endogenous	Current Power drawn at EVSE
Battery state	I_{battery}	$\mathbb{R}_{\geq 0}$	endogenous	Current power drawn at battery
	$\text{SoC}_{\text{battery}}$	$[0,1]$	endogenous	SoC of Battery
	$\hat{r}_{\text{battery}}$	$\mathbb{R}_{\geq 0}$	endogenous	Maximum charging rate at current SoC

D Additional Experiments



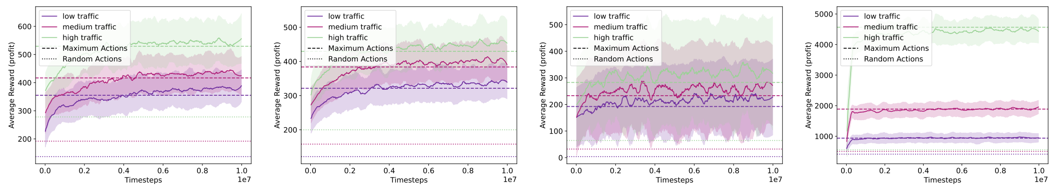
(a) Residential (b) Shopping (c) Workplace (d) Highway

Figure 6: Results on our 4 bundled scenarios using **EU** cars and 16 chargers (10 DC, 5 AC)



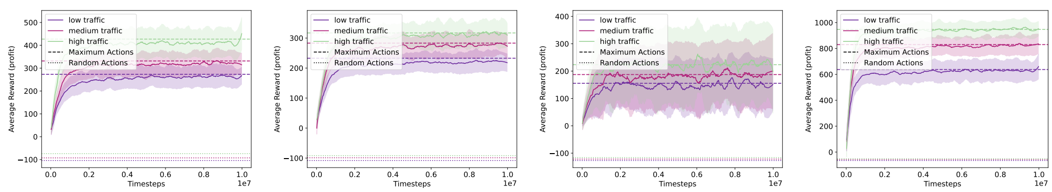
(a) Residential (b) Shopping (c) Workplace (d) Highway

Figure 7: Results on our 4 bundled scenarios using **US** cars and 16 chargers (10 DC, 5 AC)



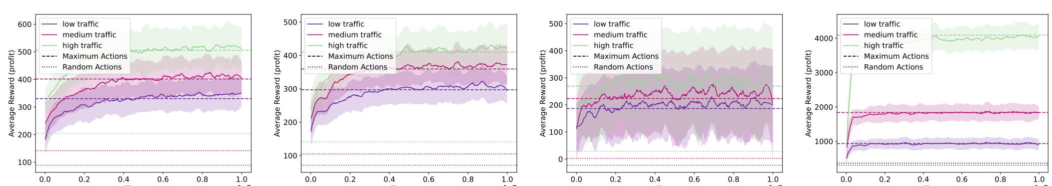
(a) Residential (b) Shopping (c) Workplace (d) Highway

Figure 8: Results on our 4 bundled scenarios using **World** cars and 16 chargers (10 DC, 5 AC)



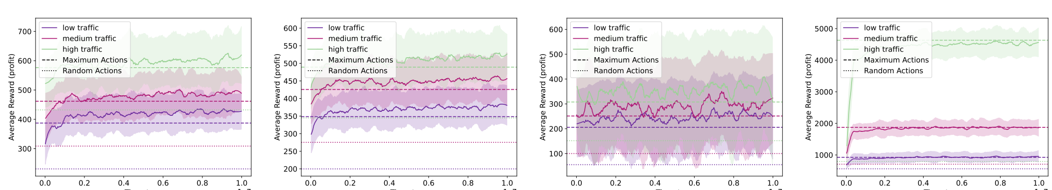
(a) Residential (b) Shopping (c) Workplace (d) Highway

Figure 9: Results on our 4 bundled scenarios using EU cars and 16 AC (11.5kW) chargers



(a) Residential (b) Shopping (c) Workplace (d) Highway

Figure 10: Results on our 4 bundled scenarios using EU cars and 8 AC (11.5kW) and 8 DC (150kW) chargers



(a) Residential (b) Shopping (c) Workplace (d) Highway

Figure 11: Results on our 4 bundled scenarios using EU cars and 16 DC (150kW) chargers