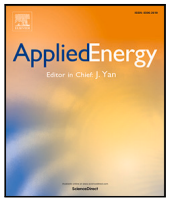
Applied Energy 301 (2021) 117504

Contents lists available at ScienceDirect

Applied Energy

journal homepage: www.elsevier.com/locate/apenergy



# Deep reinforcement learning control of electric vehicle charging in the presence of photovoltaic generation

Marina Dorokhova *, Yann Martinson, Christophe Ballif, Nicolas Wyrsch

*Photovoltaics and Thin-Film Electronics Laboratory (PV-Lab), Ecole Polytechnique Fédérale de Lausanne (EPFL), Institute of Microengineering (IMT), Rue de la Maladière 71b, 2000 Neuchâtel, Switzerland*

ARTICLE INFO

*Keywords:*
Electric vehicles
EV charging
Model-free control
PV self-consumption
Reinforcement learning
State-of-charge

ABSTRACT

In recent years, the importance of electric mobility has increased in response to climate change. The fast-growing deployment of electric vehicles (EVs) worldwide is expected to decrease transportation-related $CO_2$ emissions, facilitate the integration of renewables, and support the grid through demand–response services. Simultaneously, inadequate EV charging patterns can lead to undesirable effects in grid operation, such as high peak-loads or low self-consumption of solar electricity, thus calling for novel methods of control. This work focuses on applying deep reinforcement learning (RL) to the EV charging control problem with the objectives to increase photovoltaic self-consumption and EV state of charge at departure. Particularly, we propose mathematical formulations of environments with discrete, continuous, and parametrized action spaces and respective deep RL algorithms to resolve them. The benchmarking of the deep RL control against naive, rule-based, deterministic optimization, and model-predictive control demonstrates that the suggested methodology can produce consistent and employable EV charging strategies, while its performance holds a great promise for real-time implementations.

## 1. Introduction

### 1.1. Background and motivation

In recent years, the importance of electric mobility has increased in response to climate change, volatile prices of fossil fuels, and energy dependencies between countries. The transportation sector accounts for 27% of global greenhouse gas emissions in the EU, 72% of which are contributed by road transport [1]. Therefore, electric vehicles (EVs) are regarded as an effective way to reach emissions reduction targets and alleviate the current energy crisis. With fast-growing EV deployment around the world (+40% in 2019), where China and Europe account for 54% and 20% of the global fleet respectively [2], additional challenges arise coupled with inherent advantages of green mobility. Notably, increasing electrification of transport is expected to reshape the electricity load curve with the most pronounced effects for evening peak loads. Although EVs are unlikely to drastically drive up the overall electricity demand, the increase in peak-loads can impose significant threats on the secure and stable operation of power systems due to capacity issues of grid infrastructures. Therefore, efficient control strategies are required to manage the charging processes of EVs to avoid significant grid investments and guarantee the stability of the electricity supply. Additionally, as driving patterns demonstrate, the EVs are parked more than 80% of the time [3], which gives the potential to intelligently shift the charging load, thus deploying smart energy management techniques. On the bright side of the increasing penetration of electric mobility is the opportunity to offer grid ancillary services to support the grid's various objectives. For example, using EVs can reduce energy costs, contribute to peak shaving, improve system balancing, and integrate a larger share of renewables into power production. However, the trade-off is to combine demand–response with seamless availability of EVs, such as the primary purpose of enabling mobility is served in a reliable and timely manner.

### 1.2. Control methods

To effectively manage the charging processes of EVs, one has to choose between various control strategies. The three main broad classes of control methods include rule-based control (RBC), model predictive control (MPC), and reinforcement learning control (RLC) [4]. Indeed, there are certain advantages and disadvantages associated with each of these control techniques. Therefore, one has to choose an appropriate approach based on the trade-off between the complexity of the problem, control objectives, and available computational resources.

* Corresponding author.
*E-mail address:* marina.dorokhova@epfl.ch (M. Dorokhova).

https://doi.org/10.1016/j.apenergy.2021.117504
Received 21 January 2021; Received in revised form 20 July 2021; Accepted 28 July 2021
Available online 5 August 2021

#### 1.2.1. Rule-based control

RBC is the simplest control method that consists of a knowledge base and an inference engine. The prior defines the set of rules that govern the operations, and the latter takes actions based on the input data and the corresponding rules. Although RBC requires domain-specific expertise and knowledge to define the decision-making criteria, a rule-based nature is easy to understand as it provides transparent links between causes and effects. Therefore, solutions generated by RBC can be easily interpreted and justified. However, rule bases do not scale efficiently; thus, RBC becomes inadequate for large and complex problems. In practice, solving complicated matters with RBC results in conflicting and overlapping rules that require significant human supervision. Moreover, RBC is not good at handling incomplete or incorrect data and can strangle creativity and knowledge discovery. Indeed, only the known rules can be applied, while detecting unusual relations between the elements of the system might be hindered by the RBC nature.

#### 1.2.2. Model predictive control

MPC is a control approach based on solving a constrained optimization problem iteratively at each time step for a finite time horizon. The family of MPC methods includes deterministic and stochastic control, with the latter being able to handle uncertain disturbances without performance degradation and violation of constraints. The MPC consists of a system model, a set of constraints, an objective function, predictive trajectories of future disturbance inputs, and an optimization algorithm. The model-based nature of MPC poses significant demands to the quality and precision of the system's model while exhibiting limited generalization capabilities. Difficulties in handling nonlinear systems represent another major shortcoming of MPC. The method is not guaranteed to converge to a global optimum, nor is it guaranteed to provide a solution on time. Moreover, MPC is computationally complex, which makes it inappropriate for online use.

#### 1.2.3. Reinforcement learning control

RLC is a powerful tool for decision-making that applies to nonlinear and stochastic systems. As shown in Fig. 1, the RLC consists of an agent that interacts with the environment through actions similar to how a controller influences a technical system through control signals. Optimal control problems to be solved by RLC are formulated as Markov decision processes (MDPs). An MDP $(S, A, P, R)$ consists of the following elements: a finite set of states $S$, a finite set of actions $A$, state transition probability matrix $P$, and rewards function $R$. Therefore, the agent receives active feedback in the form of rewards when it changes the system's state by executing specific actions in the environment. The ultimate goal of the agent is to maximize the expected cumulative reward. Therefore, using reward engineering, one can define the objectives of different complexity to be achieved by the control process. Being potentially model-free is the main advantage of RLC, making it well generalizable and applicable to systems with unknown dynamics or affected by significant uncertainties. However, being a data-driven approach, RLC requires large volumes of training data, which are not always available or easy to collect from specific systems. Moreover, training demands high computational resources and long times, although a trained agent is fast and can be easily deployed online.

In the current work, we aim to explore the application of RLC to the problem of EV charging control for three main reasons. First, the fast-pace development of a connected world, thanks to the internet of things, artificial intelligence (AI), and digital transformation, coupled with increased penetration of EVs, provides conditions to generate an abundance of domain-specific data. Therefore, data-driven approaches thrive in such circumstances enabling better understanding and management of physical systems around us. Second, the model-free nature of RLC permits not only high generalization but also flexible configurations of environments. Third, extension of RLC towards deep RLC allows scalability of decision-making problems that were previously intractable. For these reasons, the following literature review focuses explicitly on the RLC methods that can be applied to the energy management of battery-powered EVs. Charging of hybrid EVs is out of the scope in the current work.

### 1.3. Related works

RLC applications to energy systems is a relatively new, though fast progressing field with the majority of works completed over the five-year preceding timeframe. Researchers in [5] conducted an extensive review of reinforcement learning (RL)-enabled demand–response, while authors in [6] looked at building energy management through the prism of RL algorithms. Both of these reviews included research on EV-related energy exchanges. In our study, we particularly highlight contributions that demonstrate a variety of objective functions, MDP formulations, and RL methods applied to the EV charging control problem. Some of the research pieces selected do not include EVs but deploy battery storage. Although availability is the crucial difference between the two, one can reuse the methodology for EV management. We divided all works into two main categories according to the action space used: discrete or continuous. We did not find any studies that deal with parametrized action spaces, which are hybrid between discrete and continuous.

Discrete action spaces are characterized by a finite number of available actions that an agent can choose from. Researchers in [7] looked at the energy storage system (ESS) and photovoltaic (PV) generation through temporal difference (TD) learning to minimize the electricity bill of residential customers. The authors in [8] applied the Q-learning algorithm, an instance of TD($\lambda$), to the energy system composed of PV, a utility grid, an ESS, a single home, and controllable home appliances. The RLC delivered a 14% reduction of electricity bill compared to the optimization approach. The same methodology was implemented in [9] with the additional inclusion of EVs and the vehicle-to-grid (V2G) concept. Another attempt at Q-learning was demonstrated in [10] to determine cost-efficient EV charging schedules with an emphasis on the battery degradation cost. However, no renewable energy sources were taken into account. Researchers in [11] deployed fitted Q-iteration batch RL to reduce the long-term cost of EV charging. Although they did not consider renewable generation, an effort was made to predict electricity prices using an artificial neural network (ANN). The action was defined by the amount of energy charged in the battery and was discretized into several equal levels ranging from 0 to battery capacity. The same algorithm in a similar problem setting was used by [12] with the demand–response goal of load flattening. The suggested MDP formulation is scalable to any number of charging stations. The authors in [13] combined the peak-shaving and the cost minimization objectives while implementing a deep Q-network (DQN) approach. Researchers in [14] pursued the same goal for a PV, EV, and building appliances energy system using DQN and deterministic policy gradient algorithms. In [15] both neural fitted Q-iteration and DQN were deployed to reduce long-term operating costs of a home energy management system coupled with ESS, PV, a utility grid, and an EV.

In a continuous action space, an action is some real-valued number usually bounded by the defined range. Often, such actions exist in multiple dimensions. The majority of the studies apply a deep deterministic policy gradient (DDPG), the most well-known RL technique to solve MDPs with continuous actions. The authors in [16] used DDPG for EV charging control to maximize user's satisfaction with charging requirements while minimizing the charging expense. Specific grid-level constraints, such as power flow, voltage region, and capacity limits of the equipment, were examined in [17] to maximize the profit of a distribution system operator in a scheme of multiple EVs. An ESS with home loads was studied in [18] to reduce total electricity costs through the actor–critic RL. However, none of these works considered renewable generation. In [19], no EVs were involved in a system of

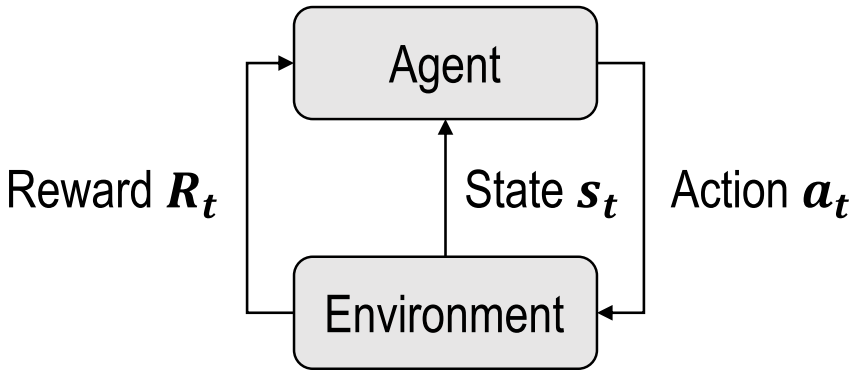



**Fig. 1.** Reinforcement learning concept.

ESS and PV generation under the energy cost minimization objective. Researchers in [20] proposed an extension of DDPG with a prioritized experience replay. Their energy system model included a utility grid, PV, ESS, heat pump, boiler, and thermal energy storage. A different approach using proximal policy optimization was demonstrated in [21] to deal with critical, shiftable, and controllable appliances, where EVs were enclosed in the latter category.

The works mentioned above were concerned with a single-agent RL, while few studies investigate the application of multi-agent reinforcement learning (MARL) approaches. The energy system in a multi-agent environment is represented by multiple actors that compete, cooperate or do both towards achieving a specific goal. Therefore, MARL finds application in the game-theoretical context. The authors in [22] considered PV, ESS, and EV charging stations to compute optimal charging schedules in a distributed manner. To reduce operational costs, they applied the CommNet algorithm on a discrete set of actions. An equilibrium-based MARL algorithm was used in [23] to determine the energy charging and discharging scheduling of a residential microgrid composed of multiple EVs and both solar and wind generation. The double objective of increasing average profits and reaching maximum autonomy from the utility grid was framed as a Markov game. The authors in [24] combined batch RL with function approximators and a market-based multi-agent system in a model-free manner. The system comprised of EVs and PV was studied under peak shaving and valley filling objectives. The results demonstrated a 50% reduction in grid's peak power. Although MARL is attractive for demand–response problems due to its distributed nature, experience sharing possibilities between agents, and potential speedup of learning using parallel computing, certain shortcomings might limit its application. Despite the curse of dimensionality problem and exploration–exploitation trade-off, which are also common for single-agent algorithms, the biggest challenge in MARL arises from the difficulty of specifying a learning goal. Moreover, the simultaneous learning of multiple agents leads to nonstationarity and an increased need for coordination between actors [25]. Thus, the methodology proposed in the current study focuses on a single agent approach.

### 1.4. Contribution

In light of the conducted literature review, we noticed that single-agent approaches prevail in EV charging control problems. Additionally, we concluded that albeit the high diversity of considered energy systems, which often include EVs, building loads, PV generation, a utility grid, and ESS, the diversification in methodology and objectives is feeble. Moreover, few works use the advent of deep RL despite its success in robotics, transportation, and healthcare. Therefore, our aim is three-fold. First, we want to fill the gap in the literature by demonstrating how the same EV charging problem can be formulated in different ways mathematically, thus triggering the application and the comparison of a variety of deep RLC algorithms. Second, we want to extend the EV control beyond the cost minimization objective as detailed below. Third, we want to show how the performance of RLC compares with other control methods such as RBC and MPC. To fulfill these goals, we focus on a simple energy system composed of a utility grid, building load, PV generation, and a single EV. Our particular contribution and the novelty of the current work lie within the following points:

- The three different mathematical formulations are proposed in the form of MDPs for the energy system of choice that enable RLC with discrete, continuous, and parametrized types of action spaces.
- The pool of EV charging control objectives is extended by focusing on maximizing PV self-consumption and EV state of charge (SOC) at departure at the same time.
- The customized OpenAI gym environments [26] are built to facilitate the development and testing of the RLC applications for the energy system of choice while being potentially exploitable for adding other energy actors, such as ESS, heat pumps, boilers, etc.
- The deep RL algorithms such as double deep Q-networks learning (DDQN), DDPG, and parametrized deep Q-networks learning (P-DQN) are applied to the problem of EV charging control. Moreover, a comprehensive benchmarking against naive, RBC, deterministic optimization, and MPC deterministic and stochastic approaches is provided.
- The application of novel techniques in the RL field, such as hindsight experience replay (HER) [27] and learning from expert demonstrations, is demonstrated to the problem of EV charging control.

The remaining of this paper is organized as follows. Section 2 describes the problem formulation, and Section 3 introduces the methodology. In Section 4, we set up the case study and present benchmark algorithms. The results are provided in Section 5 together with the discussion, while Section 6 concludes the paper with suggestions for future work.

## 2. Problem statement

The energy system in consideration consists of the following elements shown in Fig. 2. A utility grid and PV generation represent energy sources, while an EV and a building constitute energy demands. One has to note that in this work, we do not consider the V2G concept and we exploit EV as the only controllable load. In this context, we aim to control EV charging to reach two overarching objectives. Particularly, we want to increase PV self-consumption of the energy system and to achieve as high SOC at the EV departure as possible.

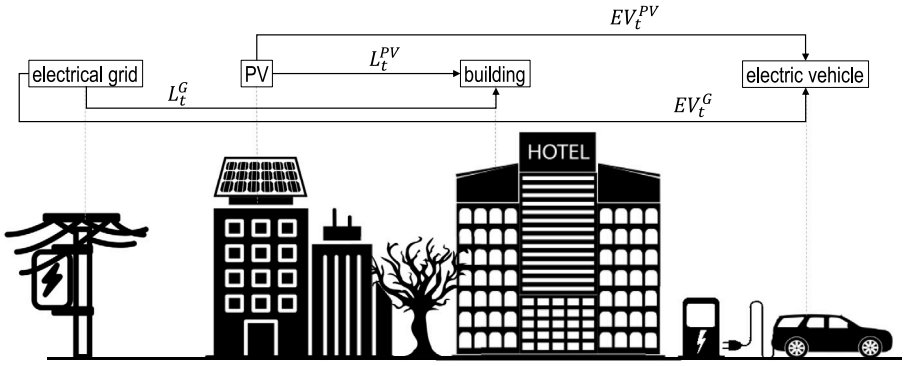



**Fig. 2.** Representation of the energy system used in the current study.

### 2.1. Objectives and constraints

Let $PV_t$ denote the PV production at time $t$, while $L_t$ and $EV_t$ are the building load demand and the EV charging demand at time $t$, respectively. The $L_t$ can be supplied by both the utility grid and PV, resulting in corresponding quantities of $L_t^G$ and $L_t^{PV}$ at time $t$, thus:

$$L_t = L_t^G + L_t^{PV} \tag{1}$$

On the contrary, according to our modeling choice, the concurrent supply to the EV from multiple sources is forbidden by the charging station:

$$EV_t^G EV_t^{PV} = 0 \tag{2}$$

where $EV_t^G$ and $EV_t^{PV}$ represent the power supplied to EV from the grid and PV, respectively. Following the definition in [28], the PV self-consumption objective can be formulated as follows:

$$\max \text{ Self-consumption} = \frac{\sum_{t=t_s}^{t_e}(L_t^{PV} + EV_t^{PV})}{\sum_{t=t_s}^{t_e} PV_t} \tag{3}$$

where the nominator is the self-consumed part of PV, while the denominator is the total PV production. One has to note that we use $t_s$ and $t_e$ to denote the corresponding start and end of the time period, which in RLC is equivalent to the length of an episode. In our problem formulation, as EV is the only controllable load, an episode starts at the arrival of EV to the charging station at $t_{arr}$ and ends with its departure at $t_{dep}$. Therefore, $t_s = t_{arr}$ and $t_e = t_{dep}$. The time resolution $\Delta t$ chosen is hourly. According to proposed notations, we formulate the maximization of the SOC at departure objective in the following way:

$$\max \ SOC_t, \text{where } t = t_{dep} \tag{4}$$

In addition to building and EV simultaneity of power supply constraints, we formulate other constraints that are necessary to respect for EV charging control problem:

$$SOC_{min} \leq SOC_t \leq SOC_{max} \tag{5}$$

$$0 \leq EV_t^G, \ EV_t^{PV} \leq P_{max} \tag{6}$$

$$SOC_{t+1} = SOC_t + \eta^{EV} \frac{(EV_t^G + EV_t^{PV})\Delta t}{C_{bat}} \tag{7}$$

where Eq. (5) bounds the EV SOC. The upper bound $SOC_{max} = 1$ is imposed by the battery capacity, and the lower bound $SOC_{min}$ is determined by the advised discharging policy. As most rechargeable batteries are not meant to be fully discharged for lifetime reasons, a minimum allowed SOC is set to avoid battery damage. In this research we assume $SOC_{min} = 0.2$, following [29]. One has to note that $SOC_t$ denotes the SOC at the beginning of time $t$. Eq. (6) imposes the upper limit on the EV charging power $P_{max}$, which is conditioned by the type of the charging station. Eq. (7) determines the continuity of the SOC within an episode, where $\eta^{EV}$ is the efficiency of the EV charging process, and $C_{bat}$ is the EV battery capacity in Wh.

### 2.2. MDP formulation

To solve the EV charging problem using RLC, one has to formulate it according to MDP formalism, thus define $S, A, P$, and $R$. As mentioned earlier, the model-free property is the advantage of RLC compared to other control methods. Therefore, we do not need to define $P$ as this would be equivalent to modeling the full dynamics of the environment. Instead, we deal with incomplete knowledge and let the next state $s'$ be determined by the environment depending on the current state $s$ and the action $a$ taken.

#### 2.2.1. State space

State space $S$ contains all possible states $s$ that an agent can have when interacting with a given environment. The system state $s$ at time $t$ is continuous and is defined as a vector $s_t = (PV_t, L_t, D_t, SOC_t)$, where in addition to PV generation, building load demand, and EV SOC, the parameter $D_t$ denotes the time to EV departure from the charging station.

#### 2.2.2. Action space

Action space $A$ includes all possible actions $a$ that an agent can perform in the environment. The definition of $A$ determines the appropriate RL algorithm to solve an MDP. In the case of EV charging control problem, we suggest three different ways to formulate $A$:

*Discrete action space.* Given the state $s_t$, action $a_t$ represents the charging power of the EV bounded by Eq. (6). In addition, $a_t$ should reflect the source of power, either grid or PV. Thus, $a_t \in \{0, \Delta e_g, \dots, \Delta e_g^{P_{max}}, \Delta e_{PV}, \dots, \Delta e_{PV}^{P_{max}}\}$, where both grid and PV power supplies are discretized according to $\Delta e_g$ and $\Delta e_{PV}$, respectively, resulting in total of 21 actions.

*Continuous action space.* The action $a_t$ is specified as a real-valued number within $[-1, 1]$ bounds. The negative part $[-1, 0)$ corresponds to the power supply from the grid, while the positive part $(0, 1]$ is equivalent to the power supply from PV. To obtain the quantity of power supply, one has to multiply the action $a_t$ by the maximum allowed power $P_{max}$ from the charging station.

*Parametrized action space.* The definition of a parametrized, discrete-continuous hybrid action space $A$, is depicted on Fig. 3. The action $a_t$ is constructed as a tuple. The discrete part of the action specifies the source, either PV or grid, while the parameter part defines the amount of power supply within a $[0, 1]$ continuous range, identical for both sources. The value 1 of a parameter is equivalent to the power supply of $P_{max}$.

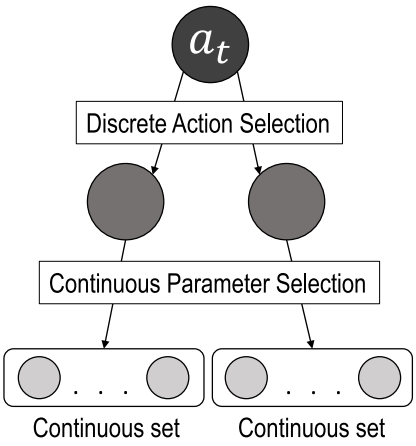



**Fig. 3.** Representation of parametrized action space.

For all the types of action spaces proposed, an additional PV production constraint has to be satisfied:

$$EV_t^{PV} \leq \max(PV_t - L_t, 0) \tag{8}$$

where the right-hand side of Eq. (8) represents the PV surplus after supplying the building load. Thus, the priority of the PV power supply is given to satisfy the $L_t$, while the EV charging from a renewable source comes secondarily.

### 2.2.3. Rewards

The reward function $R$ is the most challenging part of the MDP formulation, which usually requires careful engineering. Essentially, immediate reward $R(s, a)$ is a form of feedback the agent receives from the environment on the quality of its decision-making process. Ultimately, the shaping of the reward function $R$ should lead the agent towards achieving the objective in a fast and optimal way. Therefore, efficient reward functions facilitate the speed of convergence of RL algorithms and help the agent avoid being stuck in the local minima. The $R$ can be mainly grouped into two categories: dense and sparse. Dense rewards are given almost at every state transition, while sparse rewards are infrequent. An example of a sparse reward is a terminal reward given at the end of an episode. Dense rewards are more difficult to define as they require expertise and domain knowledge. Moreover, they are prone to limiting the agent's behavior from the discovery of creative solutions and to facilitating strange behaviors learned unintentionally [30]. Sparse rewards, instead, are easier to formulate, such as a binary signal indicating task completion or reaching the objective. However, they demand long training processes as most of the time, the agent does not receive any feedback from the environment. Additional complications arise in the case of multi-objective RLC.

*Multi-objective rewards.* In this paragraph we define the multi-objective sparse reward function $R$ that helps the agent to maximize PV self-consumption while achieving the highest SOC possible at the EV's departure. To start, we define a notion of a maximum PV self-consumption $SC_{max}$ achievable during an episode as follows:

$$SC_{max} = \frac{L_{max}^{PV} + EV_{max}^{PV}}{\sum_{t=t_s}^{t_e} PV_t} \tag{9}$$

$$L_{max}^{PV} = \sum_{t=t_s}^{t_e} \min(PV_t, L_t) \tag{10}$$

$$EV_{max}^{PV} = \min(\sum_{t=t_s}^{t_e} \max(PV_t - L_t, 0), [SOC_{max} - SOC_{t_{arr}}]C_{bat}) \tag{11}$$

Therefore, Eq. (11) is essentially a trade-off between the available PV surplus after satisfying the building demands and the available charge capacity in the EV battery. Indeed, the value of $SC_{max}$ can only be calculated retrospectively, as the RLC agent has no knowledge about future values of $PV_t$ and $L_t$. Therefore, the $R$ function should provide the rewards at the end of the episode.

To design the multi-objective $R$ we took inspiration from challenging robotics environments [31]. The goal is 2-dimensional and describes the desired PV self-consumption and SOC at the end of the episode in the form of a tuple. Therefore, the reward function $R$ can be defined as follows:

$$R = \begin{cases} 1, & if \quad (SC_{max} - SC_{t_{dep}}) \leq \epsilon_{PV} \\ & and \quad (SOC_{max} - SOC_{t_{dep}}) \leq \epsilon_{SOC} \\ 0, & otherwise \end{cases} \tag{12}$$

where $\epsilon_{SOC}$ and $\epsilon_{PV}$ are the desired tolerance values and $SC_{t_{dep}}$ is the PV self-consumption achieved at $t_{dep}$. One has to note that such a formulation of $R$ equally prioritizes the PV self-consumption and SOC at departure. In practice, however, one would give higher importance to SOC at departure as it directly impacts the EV's usability, while PV self-consumption only results in financial benefits. The sparse reward function $R$ alleviates the burden of tuning the weights in case of combining multiple objectives using a dense function. Moreover, it allows defining a unique expression for calculating rewards, which is interchangeable between discrete, continuous, and parametrized environments. On the downside, sparse $R$ requires specific techniques to speed up the training process and aid the exploration, especially when multiple objectives are combined. The HER [27] is introduced in the following section as a method that makes learning possible even if the rewards are sparse and binary.

## 3. Methodology

Following the formulation in Section 2, we have an EV charging control problem stated as an MDP with following characteristics:

- continuous state space $S$
- discrete, continuous or parametrized action space $A$
- sparse reward function $R$

In this section, we introduce RL techniques and particular algorithms that can handle the specific properties of the MDP. To create a representation of a continuous state space, we use ANNs as function approximators, which implies switching to a deep RL domain. To deal with discrete, continuous, and parametrized action spaces, we choose the DDQN, DDPG, and P-DQN model-free algorithms, respectively. To facilitate training in the environment with sparse rewards, we apply HER and learning from expert demonstrations techniques.

### 3.1. DDQN for discrete action space

The DDQN value-based algorithm [32] is meant to solve the Q-values overestimation problem of the DQN [33] by decoupling action selection from action evaluation using two separate estimators, the online network and the target network. Algorithm 1 shows the pseudo-code of the DDQN implementation. The DDQN training procedure is iterated over a specified number of episodes, where each episode lasts from $t_{arr}$ to $t_{dep}$ of the EV.

At the beginning of the algorithm, we need to create a replay buffer and to initialize online and target ANNs, where the target network is instantiated as a copy of the online network with the same weights. The first for-loop, lines 4–10, represents the interaction of the agent with the environment, where the action $a_t$ is selected using the exploration policy $\pi$ based on the current state $s_t$. The DDQN algorithm uses an $\epsilon$-greedy exploration policy, where the $\epsilon$ parameter decays to some value

**Algorithm 1** Double Deep Q-Networks Learning (DDQN)

1: *Initialize:* online network $Q_\theta$ and replay buffer $\mathcal{R}$, target network $Q_{\theta'}$ with weights $\theta' \leftarrow \theta$
2: **for** each episode **do**
3: observe current state $s_t$
4: **for** each step in the environment **do**
5: select action $a_t \sim \pi(Q_\theta(s_t))$ according to policy $\pi$
6: execute action $a_t$
7: observe next state $s_{t+1}$ and reward $r_t = R(s_t, a_t)$
8: store $(s_t, a_t, s_{t+1}, r_t)$ in replay buffer $\mathcal{R}$
9: update current state $s_t \leftarrow s_{t+1}$
10: **end for**
11: **for** each update step **do**
12: sample minibatch $N$ of experiences: $e_i = (s_i, a_i, s_{i+1}, r_i)$ from replay buffer $\mathcal{R}$
13: compute expected Q-values: $Q^*(s_i, a_i) \approx r_i + \gamma Q_{\theta'}(s_{i+1}, \arg\max_{a'} Q_\theta(s_{i+1}, a'))$
14: compute loss $L = \frac{1}{N}\sum_i (Q^*(s_i, a_i) - Q_\theta(s_i, a_i))^2$
15: perform stochastic gradient descent step on $L$
16: update target network parameters: $\theta' \leftarrow \tau\theta + (1-\tau)\theta'$
17: **end for**
18: **end for**

$\epsilon_{min}$ either linearly or exponentially. The second for-loop, lines 11–17, describes the learning procedure, enabled by the experience replay concept. The replay buffer $\mathcal{R}$ serves as memory storage that holds all transactions between the agent and the environment in the form of tuples $(s_t, a_t, s_{t+1}, r_t)$, thus allowing the agent to reuse accumulated experience for the sake of better data efficiency. When the $\mathcal{R}$ is full, the oldest experience is deleted to make space for the new one. Every $n$ steps, the minibatch $N$ of past transactions is sampled randomly from $\mathcal{R}$, thus breaking temporal relations between experiences and improving the stability of training. The learning process is three-fold. First, the expected Q-value $Q^*(s_i, a_i)$, line 13, is calculated using the Bellman equation based on the action chosen and evaluated by the online network $Q_\theta$ and the target network $Q_{\theta'}$, respectively. Second, the mean squared error loss is calculated between an expected $Q^*(s_i, a_i)$ and an actual $Q(s_i, a_i)$ Q-value and a gradient descent step performed using Adam optimizer with selected learning rate $\alpha$. Last, the parameters of the online network are slowly copied to the parameters of the target network using the rate of averaging $\tau \in (0, 1]$. In the case of hard-copying, the hyperparameter $\tau = 1$ and the update of the weights is conducted every $n$ steps.

In Table 1, we summarize our choice of hyperparameters compared to the values of the original paper [32]. For all algorithms presented in the current study, the values of hyperparameters have been determined empirically by tuning originally reported values to achieve better algorithms' performance during training. Particularly, to help the agent explore the environment, we use $\epsilon$-greedy policy with exponential decay from 0.9 to 0.1. The ANN configuration has two hidden layers of 32 nodes each with Rectified Linear Unit (ReLU) activations. The output layer size is equal to the size of the action space, and the activation is a softmax.

### 3.2. DDPG for continuous action space

The DDPG is a model-free off-policy algorithm for learning continuous actions [34]. Contrary to DQN-like algorithms, DDPG can handle continuous action spaces without discretization, thus alleviating the curse of dimensionality problem. As seen in the pseudo-code of the Algorithm 2, DDPG has an actor–critic architecture. The actor network $\mu(s|\theta^\mu)$ is the policy network that maps states to actions in a deterministic way. Thus it proposes the action to be executed. The critic network $Q(s, a|\theta^Q)$, instead, is the value network that judges the quality of the state–action pair and thus evaluates the policy. Both networks are initialized randomly. The target critic $Q'$ and the target actor $\mu'$ networks, instead, are created as copies of original networks, which can be seen in line 1. The purpose of target networks is to add stability to the training process.

**Table 1**
Hyperparameters of the DDQN algorithm.

| Hyperparameter | Symbol | Original value | Our value |
|---|---|---|---|
| Replay buffer size | $\mathcal{R}$ | $1 \times 10^6$ | $5 \times 10^5$ |
| Minibatch size | $N$ | 32 | 32 |
| Discount factor | $\gamma$ | 0.99 | 0.99 |
| Averaging rate | $\tau$ | 1 | 1 |
| Learning rate | $\alpha$ | 0.00025 | 0.001 |
| Update every | $n$ | 1000 | 32 |

**Algorithm 2** Deep Deterministic Policy Gradient (DDPG)

1: *Initialize:* critic network $Q(s, a|\theta^Q)$ and actor network $\mu(s|\theta^\mu)$, target critic network $Q'$ and target actor network $\mu'$ with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$, replay buffer $\mathcal{R}$
2: **for** each episode **do**
3: observe current state $s_t$
4: initialize random process $\mathcal{N}$ for action exploration
5: **for** each step in the environment **do**
6: select action $a_t \sim \mu(s_t|\theta^\mu) + \mathcal{N}_t$
7: execute action $a_t$
8: observe next state $s_{t+1}$ and reward $r_t = R(s_t, a_t)$
9: store $(s_t, a_t, s_{t+1}, r_t)$ in replay buffer $\mathcal{R}$
10: update current state $s_t \leftarrow s_{t+1}$
11: sample minibatch $N$ of experiences: $e_i = (s_i, a_i, s_{i+1}, r_i)$ from replay buffer $\mathcal{R}$
12: *Train critic:* compute $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$ compute loss $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$ perform stochastic gradient descent step on $L$
13: *Train actor:* $\nabla_{\theta^\mu} J \approx \frac{1}{N}\sum_i \nabla_a Q(s_i, \mu(s_i)|\theta^Q)\nabla_{\theta^\mu}\mu(s_i|\theta^\mu)$
14: update target actor and critic networks parameters: $\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$ $\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$
15: **end for**
16: **end for**

Although the interaction of the agent with the environment is similar to DDQN, the exploration in continuous action space is more challenging as the number of possible actions is infinite. The DDPG algorithm uses temporally correlated noise generated using the Ornstein–Uhlenbeck process $\mathcal{N}$ as the exploration technique [35]. In line 6, the noise value $\mathcal{N}_t$ is added to the action itself, while the process $\mathcal{N}$ is reset at the beginning of every episode. The critical difference of the DDPG algorithm from the DDQN lies in its training process as both actor and critic networks have to be trained. In line 12, the update of the critic network $Q$ is based on the Bellman equation, where Q-values of the next state $s_{i+1}$ are calculated with the target actor network $\mu'$ and the target critic network $Q'$. Afterward, mean-squared loss $L$ is minimized using Adam optimizer with a learning rate $\alpha_c$. Therefore, the critic training process is similar to the training process of the DDQN algorithm. The actor training, which is done successively, aims at maximizing the Q-values predicted by the critic based on the actions suggested by the actor itself. According to line 13, the gradients of the critic network $Q$ output computed with respect to input actions are fed back to the actor network $\mu$ using the chain-rule to update the parameters of the actor network $\mu$. Once the training is performed, the weights of the target

**Table 2**
Hyperparameters of the DDPG algorithm.

| Hyperparameter | Symbol | Original value | Our value |
|---|---|---|---|
| Replay buffer size | $\mathcal{R}$ | $1 \times 10^6$ | $5 \times 10^5$ |
| Minibatch size | $N$ | 64 | 64 |
| Discount factor | $\gamma$ | 0.99 | 0.99 |
| Averaging rate | $\tau$ | 0.001 | 0.01 |
| Learning rate actor | $\alpha_a$ | $10^{-4}$ | $10^{-4}$ |
| Learning rate critic | $\alpha_c$ | $10^{-3}$ | $10^{-3}$ |

networks $\theta^{Q'}$ and $\theta^{\mu'}$ are slowly updated based on the weights of the main networks, as seen in line 14, according to the averaging rate $\tau$.

Table 2 presents the original and empirically chosen hyperparameters of the DDPG algorithm. The Ornstein–Uhlenbeck process used for exploration is deployed with the following parameters: $\sigma = 0.2, \mu = 0, \theta = 0.15$, which are identical to those in the original paper [34]. The networks' configuration in our implementation is 32–32 for both critic and actor networks. One has to note that the learning rate of critic $\alpha_c$ is always bigger than the learning rate of the actor $\alpha_a$. Moreover, a previous study [36] on the RL for EV management has indicated that the DDPG algorithm is susceptible to the choice of hyperparameters and frequently, even a single wrongly chosen parameter can sabotage the learning process.

### 3.3. P-DQN for parametrized action space

Parametrized action spaces are usually either discretized into approximate finite sets or relaxed into continuous sets. However, several attempts were made to develop RL algorithms specifically for hybrid action spaces. These works include Q-PAMDP algorithm presented in [37], hybrid actor–critic architecture demonstrated in [38], extended DDPG shown in [39], and the P-DQN algorithm suggested in [40]. We have chosen the latter algorithm for solving the parametrized EV charging control problem due to its intuitive implementation, as it encompasses the properties of both DQN and DDPG techniques and off-policy nature.

Algorithm 3 describes the P-DQN implementation according to [41], where the original methodology was enhanced with target networks to improve stability during training. The action $a_t$ in parametrized action space $A$ is defined as a tuple $(k_t, x_{k_t})$. The prior $k_t \in K$ represents the discrete action from the set of all discrete actions $K$, while the latter $x_{k_t}$ represents the continuous parameters associated with this specific discrete action. The P-DQN algorithm is based on two main networks: action value network $Q$ and action parameter network $\mu$. These networks, together with their target copies and a replay buffer $\mathcal{R}$, are initialized in line 1. The $\mu$ network takes the current state $s_t$ as input and produces the parameters $x_k$ for all actions $k \in K$. The $Q$ network outputs the Q-values for all actions $k$ while taking the state $s_t$ and the parameters $x_k$ as an input. Afterward, the desired action $a_t$ is determined by the $\epsilon$-greedy policy, as it can be seen in line 8. Notably, the P-DQN algorithm deploys two different exploration techniques. In addition to the $\epsilon$-greedy that helps explore the discrete part of the action space, the noise process $\mathcal{N}$ is used to explore the continuous part. In our implementation, we use Ornstein–Uhlenbeck process noise, similar to the DDPG algorithm. Lines 14 and 15 show the training process of the action value $Q$, and the action parameter $\mu$ networks, which is similar to the training process of the DDPG. Once the training is complete, the weights are slowly copied from the main networks to target networks $Q'$ and $\mu'$, as depicted in line 16.

The hyperparameters are collected in Table 3. The network configurations are 32–32 and 32–32 for the action value $Q$ and the action parameter $\mu$ networks, respectively. The parameters of the Ornstein–Uhlenbeck exploration process are identical to those of the DDPG algorithm. The $\epsilon$-greedy exploration process starts with $\epsilon$ value of 0.5 and decays exponentially until 0.01.

**Algorithm 3** Parametrized Deep Q-Networks Learning (P-DQN)

1: *Initialize:* action value network $Q(s, x_k|\theta^Q)$ and action parameter network $\mu(s|\theta^\mu)$, target action value network $Q'$ and target action parameter network $\mu'$ with weights $\theta^{Q'} \leftarrow \theta^Q, \theta^{\mu'} \leftarrow \theta^\mu$, replay buffer $\mathcal{R}$
2: **for** each episode **do**
3: observe current state $s_t$
4: initialize random process $\mathcal{N}$ for action parameter exploration
5: **for** each step in the environment **do**
6: compute action parameters $x_k \leftarrow \mu(s_t|\theta^\mu) + \mathcal{N}_k$
7: compute action values $Q_k \leftarrow Q(s_t, x_k|\theta^Q)$
8: select action $a_t = (k_t, x_{k_t})$ according to the $\epsilon$-greedy policy
9: execute action $a_t$
10: observe next state $s_{t+1}$ and reward $r_t = R(s_t, a_t)$
11: store $(s_t, a_t, s_{t+1}, r_t)$ in replay buffer $\mathcal{R}$
12: update current state $s_t \leftarrow s_{t+1}$
13: sample minibatch $N$ of experiences:
$e_i = (s_i, a_i, s_{i+1}, r_i)$ from replay buffer $\mathcal{R}$
decompose $a_i$ into $k_i$ and $x_{k_i}$
14: *Train Q:*
compute $y_i = r_i + \gamma \max_{k_i \in [K]} Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
compute loss $L = \frac{1}{N}\sum_i (y_i - Q(s_i, x_{k_i}|\theta^Q))^2$
perform stochastic gradient descent step on $L$
15: *Train μ:*
compute loss $L = -\frac{1}{N}\sum_i Q(s_i, \mu(s_i|\theta^\mu)|\theta^Q)$
perform stochastic gradient descent step on $L$
16: update target action value and target action parameter networks parameters:
$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$
$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$
17: **end for**
18: **end for**

**Table 3**
Hyperparameters of the P-DQN algorithm.

| Hyperparameter | Symbol | Original value | Our value |
|---|---|---|---|
| Replay buffer size | $\mathcal{R}$ | $1 \times 10^6$ | $5 \times 10^5$ |
| Minibatch size | $N$ | 64 | 32 |
| Discount factor | $\gamma$ | 0.99 | 0.99 |
| Averaging rate | $\tau$ | 0.01 | 0.001 |
| Learning rate action value | $\alpha_a$ | $10^{-4}$ | $10^{-3}$ |
| Learning rate action param | $\alpha_{ap}$ | $10^{-5}$ | $10^{-4}$ |

### 3.4. Hindsight experience replay (HER)

To enable sample-efficient learning from sparse and binary rewards, we apply the HER technique developed in [27]. On the one hand, unlike humans, the model-free off-policy algorithms presented have difficulties learning from failures. On the other hand, the sparse reward structure makes successes rare, thus complicating the training process. The HER technique aims to extract useful information from past environment transactions using the notion of goals. At the beginning of each episode, the desired goal $g_d$ is sampled along with the observation of the current state $s_t$. In our case, $g_d$ can be the requested SOC at the EV departure, the desired level of PV self-consumption, or both if the $g_d$ is formulated in the form of a tuple. The whole episode $s_{t_0}, s_{t_1}, \ldots, s_T$, where $s_T$ is the state at the episode's terminal, is stored in the form of experiences $s_t, a_t, s_{t+1}, r_t$ in the transaction buffer $\mathcal{D}$. At the end of the episode, the agent arrives to some achieved goal $g_a$, which can be identical to $g_d$, if the episode was a success, or can be different if it was a failure. Here the HER technique comes into play to efficiently utilize the experiences of the episode even if $g_d$ was not achieved. We use the information stored in the $\mathcal{D}$ buffer to replay the whole episode pretending that the achieved goal $g_a$ was our desired goal from the

beginning. Therefore, our failure episode becomes a success episode, where the agent acquires the knowledge, learns to achieve the goal $g_a$, and obtains the reward for successful completion of the task. The newly replayed episode is stored in the replay buffer $\mathcal{R}$, where it serves as a positive example to train the agent. In our implementation of the HER technique that follows the original paper [27], we used the strategy *final*, which compares the achieved and desired goals at the terminal state of the episode. The addition of HER technique requires the specific implementation of the environment where the goal is concatenated to the state. More details on how one can achieve it in practice are given in Section 4.4.

### 3.5. Expert demonstrations

The inclusion of expert demonstrations is another technique suited for facilitating the training process by improving the exploration of the environments with sparse rewards. In our work, we aim to use expert demonstrations to enhance the learning and speed up the convergence of the DDPG algorithm in particular. The main reasons include the tedious exploration of infinite continuous action space and the difficulty of tuning the DDPG algorithm's hyperparameters, which were previously mentioned in Section 3.2. Moreover, previous successful implementations of the DDPG algorithm leveraging expert demonstrations in robotics [30,42] and in the field of autonomous driving [43] motivate our choice.

The main idea to combine expert demonstrations with RL is to aid the agent's training by providing experiences where the outcome was successful. The practical implementation of DDPG with expert demonstrations consists of three main steps. First, the second replay buffer $\mathcal{R}_D$ is created to store the demonstration data in addition to the original replay buffer $\mathcal{R}$ that holds the self-generated data. Both data are stored in the form of experience tuples $(s_i, a_i, s_{i+1}, r_i)$. Second, during the training, the minibatch $N$ of previous transactions is sampled from both replay buffers $\mathcal{R}_D$ and $\mathcal{R}$ according to the predefined proportion. Third, the loss $L$ computed during the actor's training is augmented with the behavior cloning loss $L_{BC}$ according to [42]. The additional loss is applied only on demonstration samples and is calculated as follows:

$$L_{BC} = \sum_{i=1}^{N_D} \|(\mu(s_i)|\theta^\mu) - a_i\|^2 \tag{13}$$

where $N_D$ is the amount of demonstration samples in the minibatch $N$, $a_i$ is the action taken by the expert and $\mu(s_i)|\theta^\mu$ is the action produced by the agent's policy. Therefore, behavior cloning loss $L_{BC}$ is combined with the loss of the actor using $\lambda_1$ and $\lambda_2$ weights:

$$L = \lambda_1 \nabla_{\theta^\mu} J - \lambda_2 \nabla_{\theta^\mu} L_{BC} \tag{14}$$

## 4. Case study

To validate the proposed methods for solving the EV charging control problem, we created a case study based on the green e-mobility project [44] conducted within the research framework of the SCCER FURIES [45]. The project, established as part of activities related to the Swiss National Action Plan on Digitalization [46], integrates a network of EVs and charging stations available to guests staying in the hotels of Val d'Hérens alpine region in Switzerland. Each of the eight partner hotels owns one charging station and at least one EV, allowing guests to explore the region with maximum independence and minimum harm to the environment. The EVs, the majority of them being a Citroen C-Zero model, are rented to the hotels' guests daily free of charge.

The agent is an EV defined by its battery capacity and the upper limit on the charging power $P_{max}$. In our case study, the Citroen C-Zero model of the EV has 16 kWh battery and 3700 W maximum power input [47]. In addition, to simplify the calculations, we assume the efficiency of the EV charging process $\eta^{EV} = 1$.

### 4.1. Datasets

The test dataset was collected from one of the hotels participating in the project. The dataset spans over a period of 2 months from the 9th of September to the 9th of November 2020. The building load demand $L_t$ and the PV production $PV_t$ measurements were down-sampled to hourly resolution. Due to the absence of the real-world recordings of the EV-related part of the dataset, we simulated EV usage patterns. The SOC at arrival, time of arrival at the charging station, and time of departure from the charging station were sampled from uniform distributions in the following way:

- $SOC(t = t_{arr}) \sim \mathcal{U}(0.2, 0.5)$
- $t_{dep} \sim \mathcal{U}(7, 19)$
- $t_{arr} \sim \mathcal{U}(t_{dep} + 1, 23)$

According to chosen distribution boundaries, we assume a minimum trip duration of one hour and EV return to the hotel on the day of departure. The time to EV departure from charging station $D_t$, which is necessary for defining the state $s_t$, was computed as the number of hours between the EV arrival at $t_{arr}$ and the next departure $t_{dep}$.

Due to the limited availability of data from the hotel, the training dataset for RLC algorithms was synthesized based on another hotel in the area. The smart meter measurements were scaled to reflect the test hotel's size, while the PV production of a virtual installation was simulated using relevant recordings from a nearby meteorological station. The EV driving scenario was generated using the same uniform distributions as shown above. The length of a training dataset is one year.

### 4.2. Performance evaluation

The performance of DDQN, DDPG, and P-DQN algorithms is evaluated in two phases. First, we present the results obtained during training in Section 5.1. Following the established goals, we report both the PV self-consumption (Eq. (3)) and SOC at departure (Eq. (4)) metrics. To enhance comprehension of the results, the optimal PV self-consumption is computed, according to Eq. (9). The addition of this metric gives better judgment on the PV usage by the EV charging process. To ensure fairness of reporting deep RL algorithms' performance, we use the guidelines indicated in [48]. Thus, we run five experiments for each of the algorithms using the same set of random seeds. Further, we summarize the DDQN, DDPG, and P-DQN algorithms' performance by reporting the evolution of mean and standard deviation throughout the training process for each of the objectives across five seeds. Moreover, to provide judgment on the complexity of suggested deep RL algorithms and the computational resources required, we report the duration of the training process.

In the second phase, both benchmark and deep RLC algorithms are executed on the held-out test set of historical data consisting of 60 episodes. For all algorithms we report the performance metrics, namely PV self-consumption and SOC at departure, in the form of a box-plot visualization that provides a five-number summary: the minimum, the maximum, the sample median, and the first and the third quartiles across 60 episodes. Moreover, we report the amount of energy purchased from the grid in the same format along with its distribution. To highlight the differences in the EV charging approaches suggested by the algorithms, we additionally compare the evolution of the SOC across episodes with the same SOC at arrival and across episodes with binned duration. Lastly, we give information on the execution time of the algorithms on the test set to understand the suitability for online implementations.

### 4.3. Benchmark algorithms

To understand the quality of the solutions provided by RLC methods, we compare them among themselves and other classic approaches used in the EV charging control problem. In this section, we present several algorithms used to benchmark the proposed methodology. Importantly, all benchmark solutions have to respect the SOC (Eq. (5)) and the power limit (Eq. (6)) constraints introduced in Section 2.

#### 4.3.1. Naive EV charging

The naive algorithm reflects the conventional charging attitude of the majority of EV drivers nowadays. The EV starts to charge as soon as it is plugged to the charging station at $t_{arr}$. If there is any PV surplus, the EV draws the available PV power $EV_t^{PV}$, otherwise the power supply $EV_t^G$ comes from the grid and is equal to $P_{max}$.

#### 4.3.2. RBC EV charging

Whenever there is an excess of PV production, the EV uses the renewable power to charge. If the PV is not available, the algorithm evaluates the possibility of drawing power from the grid based on the EV's $SOC_t$ and the time left before departure from the charging station $D_t$. Based on the following expression, the EV compares the leftover energy to be charged in the battery with the maximum possible transmittable energy during the remaining time.

$$(SOC_{max} - SOC_t)C_{bat} \leq (D_t - t_{lag})P_{max} \tag{15}$$

The time parameter $t_{lag}$ tunes how shortsighted the algorithm is. A large value of $t_{lag}$ helps the control to anticipate the EV's departure, however, it should not exceed the time required to fully charge the EV from $SOC_{min}$ to $SOC_{max}$ using $P_{max}$ from the grid. As a result, the EV draws $P_{max}$ from the grid at time $t$ if the EV departs rather soon and chooses not to charge otherwise.

#### 4.3.3. Deterministic optimization EV charging

The deterministic optimization of EV charging, aimed to provide the optimal solution, is formulated with the following objective function that encompasses both, maximization of the PV self-consumption and SOC at departure (where $k_1$ and $k_2$ are respective weights for the two objectives, such as $k_1 + k_2 = 1$):

$$\max\left(k_1\left(\frac{\sum_{t=t_s}^{t_e}(EV_t^{PV} + L_t^{PV})}{\sum_{t=t_s}^{t_e} PV_t}\right) - k_2\left(SOC_{max} - SOC_{t_{dep}}\right)\right) \tag{16}$$

subject to constraints for $t = t_s, \ldots, t_e$:

- ▷ $PV_t + y_t^G P_t^B - (1 - y_t^G)P_t^S - EV_t - L_t = 0$,
  where $P_t^B$ is the power bought from the grid, $P_t^S$ is the power sold back to the grid, and $y_t^G \in \{0, 1\}$ is the binary variable that forbids the simultaneous purchasing and selling of the power from and to the grid
- ▷ $PV_t = EV_t^{PV} + L_t^{PV} + P_t^S$
- ▷ $P_t^B = EV_t^G + L_t^G$
- ▷ $EV_t = y_t^{EV} EV_t^G + (1 - y_t^{EV})EV_t^{PV}$,
  where $y_t^{EV} \in \{0, 1\}$ is the binary variable that governs the simultaneity of power supply to EV according to Eq. (2), such as $y_t^{EV} \leq y_t^G$
- ▷ $SOC_{t_{arr}} = SOC_{arr}$
- ▷ Load satisfaction constraint (Eq. (1))
- ▷ SOC limits (Eq. (5))
- ▷ EV charging power limit (Eq. (6))
- ▷ SOC continuity (Eq. (7))

#### 4.3.4. Deterministic MPC EV charging

The deterministic formulation of MPC is iterative, thus the algorithm takes charging decisions at every time step $t$. The episode $t_s$ to $t_e$ is separated into two successive stages: the decision stage , where the actual time step $t = t_d$, and the prediction stage $t_f = t_{d+1}$ to $t_e$. At the decision stage the PV production $PV_{t_d}$ and the load consumption $L_{t_d}$ values are known, while at the prediction stage the Long Short-Term Memory (LSTM) neural network and the Auto Regressive Integrated Moving Average (ARIMA) model are used to predict the $PV_T$ and $L_T$ values for $T = t_f, \ldots, t_e$ respectively. Therefore, the objective function, though similar to Eq. (16), is transformed in the following way to reflect the two-stage formulation:

$$\max\left(k_1\left(\frac{EV_{t_d}^{PV} + L_{t_d}^{PV}}{PV_{t_d}} + \frac{\sum_{t=t_f}^{t_e}(EV_t^{PV} + L_t^{PV})}{\sum_{t=t_f}^{t_e} PV_t}\right) - k_2\left(SOC_{max} - SOC_{t_{dep}}\right)\right) \tag{17}$$

The constraints applied to the model are identical to those for deterministic optimization stated in Section 4.3.3. However, all constraints are formulated for both decision and prediction stages.

#### 4.3.5. Stochastic MPC EV charging

The stochastic formulation of MPC differs from the deterministic formulation during the prediction stage, as LSTM and ARIMA models forecast confidence intervals for $PV_T$ and $L_T$. Therefore, one can draw an $\Omega$ set of $N$ scenarios from predicted distribution, where each scenario $w_i \in \Omega$ consists of $PV_T^{w_i}$ and $L_T^{w_i}$ forecast values. All constraints stated in Section 4.3.3 hold true for the stochastic MPC and require two-stage formulation. To take into account all possible scenarios from $\Omega$ set, the objective function has to be reformulated in the following way, assuming that each $w_i \in \Omega$ scenario is equally probable:

$$\max\left(k_1\left(\frac{EV_{t_d}^{PV} + L_{t_d}^{PV}}{PV_{t_d}} + \frac{1}{N}\sum_{i=0}^{N}\frac{\sum_{t=t_f}^{t_e}(EV_t^{PV,w_i} + L_t^{PV,w_i})}{\sum_{t=t_f}^{t_e} PV_t^{w_i}}\right) - k_2\frac{1}{N}\sum_{i=0}^{N}\left(SOC_{max} - SOC_{t_{dep}}^{w_i}\right)\right) \tag{18}$$

### 4.4. Implementation

To apply RLC for the EV charging control problem, we implemented both the environments and the algorithms using Python programming language. Custom discrete, continuous, and parametrized environments were created using the OpenAI Gym toolkit [26]. A specific *gym.GoalEnv* class was chosen to build goal-based environments to enable the HER technique. The RL algorithms were developed using *tensorflow 2.0* library. The deterministic optimization and both stochastic and deterministic MPC algorithms were implemented using *Gurobi* solver. All the experiments were conducted using a personal laptop (Intel i7- 7600, 16 GB RAM).

## 5. Results

### 5.1. Training phase

Fig. 4 depicts DDQN, DDPG, and P-DQN algorithms' performance during the training phase of 2000 episodes on a one-year dataset. All algorithms deploy the HER technique to facilitate the training. The DDPG algorithm's learning process is also enhanced by 100 episodes of expert demonstrations.

The bold line and the shaded region on each subplot show the mean and the standard deviation of 5 runs, each corresponding to a different seed. The evaluation of algorithms with various seeds is important to understand the models' robustness, validate the choice of the hyperparameters, and compare the algorithms among themselves.

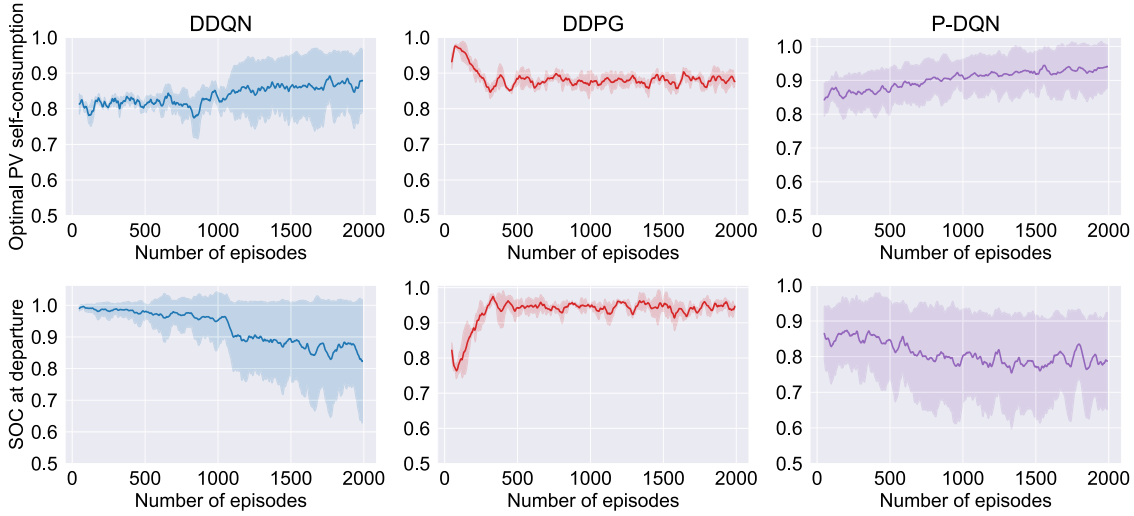



**Fig. 4.** Training performance of deep reinforcement learning algorithms.

Although seeding can fix the random processes to get reproducible results in controlled experiments, one has to note that randomness is an inherent feature of machine learning models. Therefore, when commissioning models in production, one should remove the fixed seed value. During the training process, we have identified the following major parts of the reinforcement learning workflow that introduce randomness:

- Arrival SOC initialization at the beginning of each episode
- Weight initialization in neural networks
- Adam optimizer
- Ornstein–Uhlenbeck and $\epsilon$-greedy exploration policies
- Minibatch sampling from replay buffer $\mathcal{R}$

As shown in Fig. 4, the P-DQN algorithm converges to higher optimal PV self-consumption, followed by DDPG and DDQN, with a substantial share of episodes reaching the maximum PV self-consumption possible. The same metric for DDPG stabilizes to 0.9 with a very low standard deviation, which can be explained by the inclusion of behavior cloning loss from expert demonstrations along the training process. The DDQN algorithm shows interesting behavior, as PV self-consumption significantly changes its variance and starts to fluctuate between 0.8 and 0.95 from approximately 1100 episodes. The turning point corresponds to the $\epsilon$-value of 0.3 in the $\epsilon$-greedy policy, which decays exponentially throughout the training process. The same behavior of the DDQN algorithm can be noted for the SOC at departure metric. The standard deviation across runs increases, while the mean SOC at departure decays to an average of 85%. However, during the whole training process, a substantial share of the episodes achieves $SOC_{max}$ at departure. The P-DQN algorithm exhibits high variance as well, with the mean SOC at departure at 80%. The DDPG algorithm converges to the highest SOC at departure of 95% among all algorithms with very low variance. Moreover, the DDPG demonstrates the fastest speed of learning due to facilitation through expert demonstrations. The training results overall confirm that deep RLC algorithms are capable of achieving multiple objectives simultaneously. The maximization of the PV self-consumption is more pronounced for DDQN and P-DQN algorithms, although the SOC at departure values show a decaying trend. However, the mean values of SOC at departure for these algorithms are exceeding the 80% limit, which is considered acceptable SOC before departing for the majority of EV drivers.

Fig. 5 depicts the training duration of deep RLC algorithms. The DDQN algorithm is shown twice as the averaging rate $\tau$ and the update every $n$ hyperparameters influence the training duration differently. With $\tau = 0.01$, the online neural networks are trained at every step

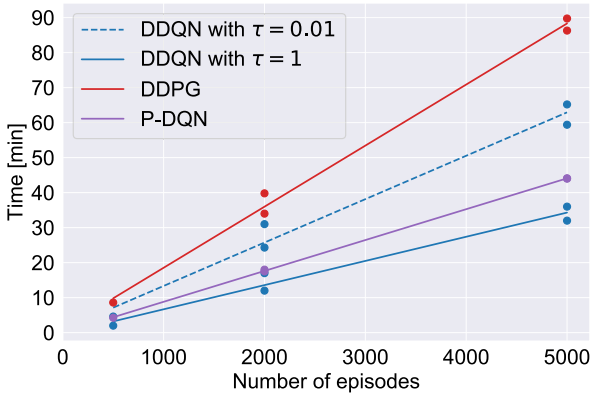



**Fig. 5.** Training duration of reinforcement learning algorithms.

$n = 1$, and the weights are slowly copied to the target networks. With $\tau = 1$, the training is performed every $n = 32$ steps; thus, the overall process takes considerably less time.

There are three key observations that one can make from Fig. 5. First, the training duration is the linear function of the number of episodes, which is useful to estimate required time resources before engaging in training. Second, the execution time varies for the same number of episodes for the same algorithm as each episode's duration is different, and the environment is reset at the beginning of each episode. Third, the DDPG algorithm requires almost twice longer training time than other algorithms due to the complexity of its neural networks framework, larger minibatch size, and additional computation of behavior cloning loss.

### 5.2. Testing phase

During the testing phase, the deep RLC algorithms are compared with benchmark algorithms chosen in Section 4.3 on a held-out test dataset of 60 episodes. The current section presents the results in ascending order of time granularity, starting with observations per episode and following with comparison across the whole test dataset.

Fig. 6 depicts the EV charging strategies for all algorithms on three sample episodes. The episodes were chosen to demonstrate the diversity of PV production and load consumption profiles. The deterministic algorithm demonstrates the optimal performance in terms of both, the amount of PV consumed for EV charging and the $SOC_{t_{dep}}$. Despite the


**Fig. 6.** Examples of benchmark and RLC algorithms on selected episodes of test data.

Naive algorithm always supplying the grid power for EV charging, as the arrival times of the EV are outside the PV production hours, the algorithm can guarantee the $SOC_{t_{dep}} = SOC_{max}$. The RBC algorithm exhibits the charging strategy similar to the optimal performance, though some of the grid power inputs are shifted in time. Both deterministic and stochastic MPC algorithms show good usage of PV generation and achieve rather high $SOC_{t_{dep}}$ for all three episodes. However, the sudden load consumption peaks, such as at the end of the episode 1, are not handled well to provide $SOC_{max}$ at departure. The deep RLC algorithms demonstrate tendency towards using the most PV power possible and in two out of three episodes reach high $SOC_{t_{dep}}$. However, the absence of the PV power production period in the episode 3 prevents all RLC algorithms from ending the episode with high SOC, while the short duration of the episode leaves less time for decision-making. The DDPG algorithm demonstrates a different grid power supply strategy from all other algorithms. Particularly, the DDPG takes lower risks by purchasing electricity early and in small quantities, thus anticipating potential future disturbances in PV power production.

The boxplots in Fig. 7 summarize the algorithms' performance across all episodes in the test dataset. The deterministic algorithm, indeed, serves as the baseline for comparison as it provides the optimal performance. The Naive algorithm demonstrates the lowest PV self-consumption values among algorithms while it guarantees the $SOC_{t_{dep}} = SOC_{max}$. The RBC algorithm performs closely to the optimal solution, though with almost twice higher variance in PV self-consumption than the deterministic algorithm. The stochastic MPC outperforms the deterministic MPC in PV self-consumption, while in some cases it does not reach the $SOC_{max}$ at departure. The deep RLC algorithms vary significantly in their $SOC_{t_{dep}}$ levels from all other algorithms, with the DDPG having the highest $SOC_{t_{dep}}$ values among RLC algorithms. The DDQN algorithm demonstrates mediocre performance for both objectives. The P-DQN algorithm achieves a higher median for PV self-consumption than for $SOC_{t_{dep}}$; however, the algorithm is characterized by very high variance across episodes. The median values of energy purchased from the grid are very similar among all algorithms, with DDQN and DDPG using less grid energy. However, marginal increases of the purchased energy amounts for these algorithms would possibly compensate for lower $SOC_{t_{dep}}$. Therefore, one can conclude that major discrepancies between EV charging strategies result primarily from differences in the usage of PV generation.

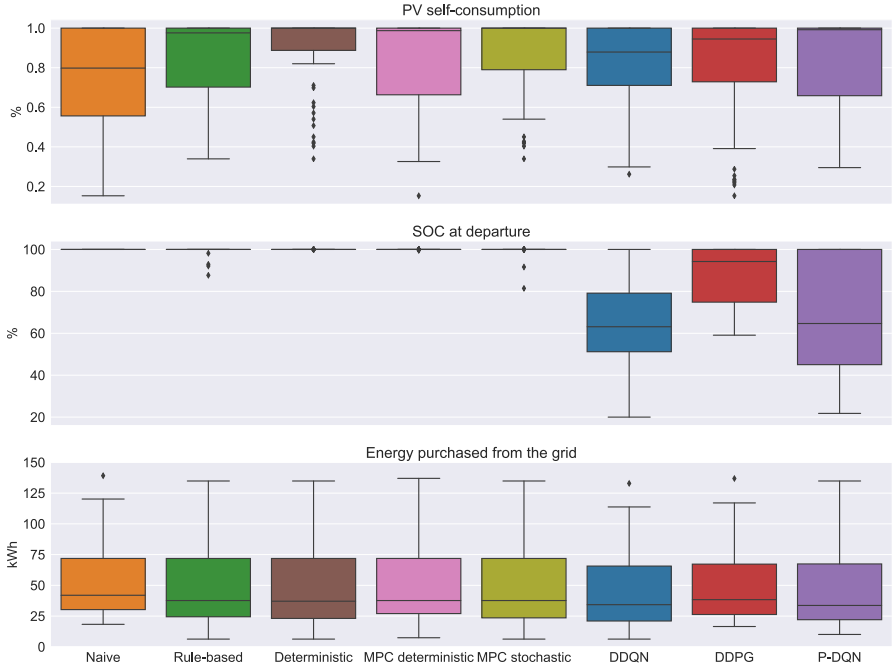



**Fig. 7.** RLC and benchmark algorithms' comparison across episodes.

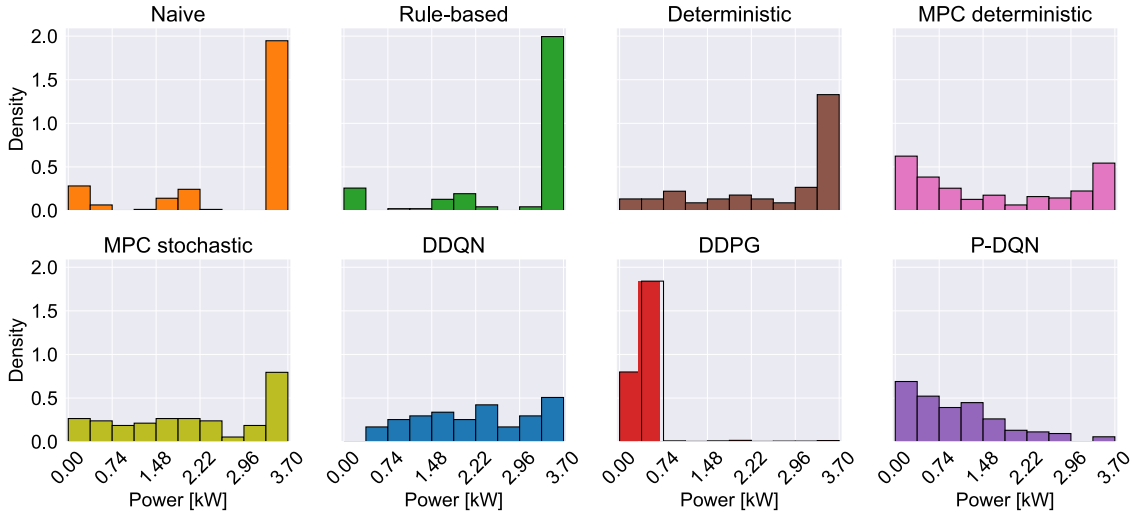



**Fig. 8.** Comparison of grid power usage across algorithms.

Fig. 8 shows density histogram plots of grid power usage for EV charging among the algorithms. Although the flattening of the grid-supplied part of the EV charging curve was not specified as one of the control objectives, we can observe the difference in grid power utilization among algorithms. Particularly, one can notice that both Naive and RBC algorithms rely heavily on $P_{max}$ injections to the EV. Instead, the power of smaller magnitudes is used occasionally and is more probable to appear closer to the end of the charging process as the SOC increases. The deterministic and the stochastic MPC have similar grid power usage patterns, with the prior supplying higher power more frequently. On the contrary, the deterministic MPC has a more balanced profile of EV supply from the grid, with minimal and substantial power values being equally probable. Although the DDQN's grid power usage pattern is similar to that of non-RLC algorithms, the DDPG and P-DQN demonstrate different approaches to drawing power from the grid for EV charging. Particularly, the DDPG algorithm relies only on small and frequent grid power quantities and does not induce any high power peaks during the charging process. Such behavior can be explained by the DDPG's anticipation and risk minimization strategy. Similarly, the P-DQN algorithm uses small-magnitude grid power values while rarely injecting $P_{max}$ to the EV.

Fig. 9 depicts the SOC evolution across episodes with the same $SOC_{t_{arr}} \in \{20\%, 30\%, 40\%, 50\%\}$, where the speed of SOC changing and the differences between the charging approaches undertaken by considered algorithms can be observed. The lengths of the episodes were normalized to provide the possibility to compare the SOC profiles across charging processes with various duration. The Naive algorithm shows a steep profile of SOC evolution reaching $SOC_{max}$ within initial 20% of the charging time regardless of the $SOC_{arr}$. Instead, the RBC algorithm starts to charge the EV closer to the end of the charging process. The deterministic approach and both MPC algorithms have less steep SOC evolution profiles, with the charging process starting later for EVs arriving with $SOC_{arr} = 50\%$ than for EVs with lower $SOC_{arr}$. However, one can notice that with $SOC_{arr} = 30\%$ almost the

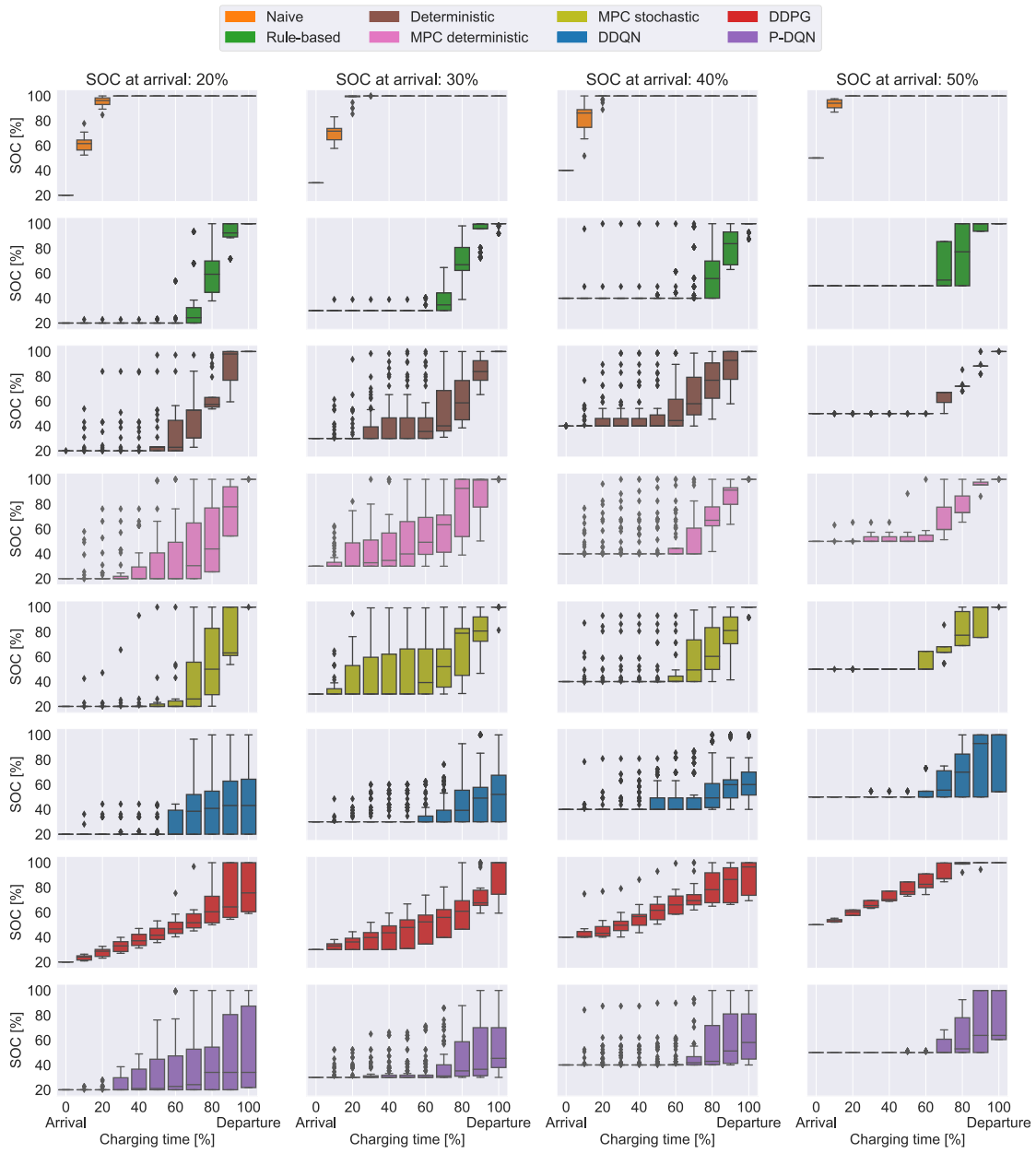



**Fig. 9.** Comparison of SOC evolution across episodes with equal SOC at arrival.

whole duration of the episode is used to charge the EV. The DDPG algorithm stands out with its linear SOC profile and effective usage of the whole episode for charging. Moreover, one can notice that the cases of not achieving the $SOC_{max}$ at departure are more common for $SOC_{arr} = 20\%$. The P-DQN algorithm increases its waiting time before charging according to $SOC_{arr}$ increases, while the same feature does not appear in the DDQN's SOC evolution profile.

Fig. 10 shows the SOC evolution, PV generation, and load demand profiles across episodes with binned duration. One can notice that longer episodes feature two peaks of PV generation and load consumption. While the EV charging strategies of Naive and RBC algorithms do not exhibit noticeable changes with increased episode's length, the deterministic and both MPC algorithms demonstrate interesting changes in the SOC profile. The variance during the initial 20% of the charging time can be explained by the spread of the $SOC_{arr}$. The SOC profiles for short episodes below 14 h exhibit logarithmic growth, while the episodes with 19–26-hour duration show exponential growth. The latter can be explained by weak PV generation profiles during the initial 50% of the charging time. The shape of the SOC curve for long episodes changes to accommodate a double-period of PV generation; thus, the SOC increases initially and at the end of an episode, with the middle part being relatively steady. A similar double-curvature feature can be noticed for the P-DQN algorithm at very long episodes, while everywhere else, the SOC profile resembles a mixture between linear and exponential. The DDPG algorithm demonstrates a stable linear charging trend regardless of the episode's duration. The DDQN's SOC profile shows efficient usage of PV generation for episodes with 23–26-hour duration. However, for other episodes, the small slope of rather a linear trends results in low $SOC_{dep}$.

After looking at the algorithms' performance across episodes, we summarize the results on the whole test dataset in the remainder of this section. Table 4 expresses the SOC at departure satisfaction levels for each algorithm. The low level of satisfaction corresponds to $SOC_{dep} \leq 50\%$, which makes the EV unsuitable for usage by the hotel's guests. The high level of satisfaction conforms with $SOC_{dep} \geq 80\%$ and is achieved by all non-RLC algorithms on 100% of test episodes. The

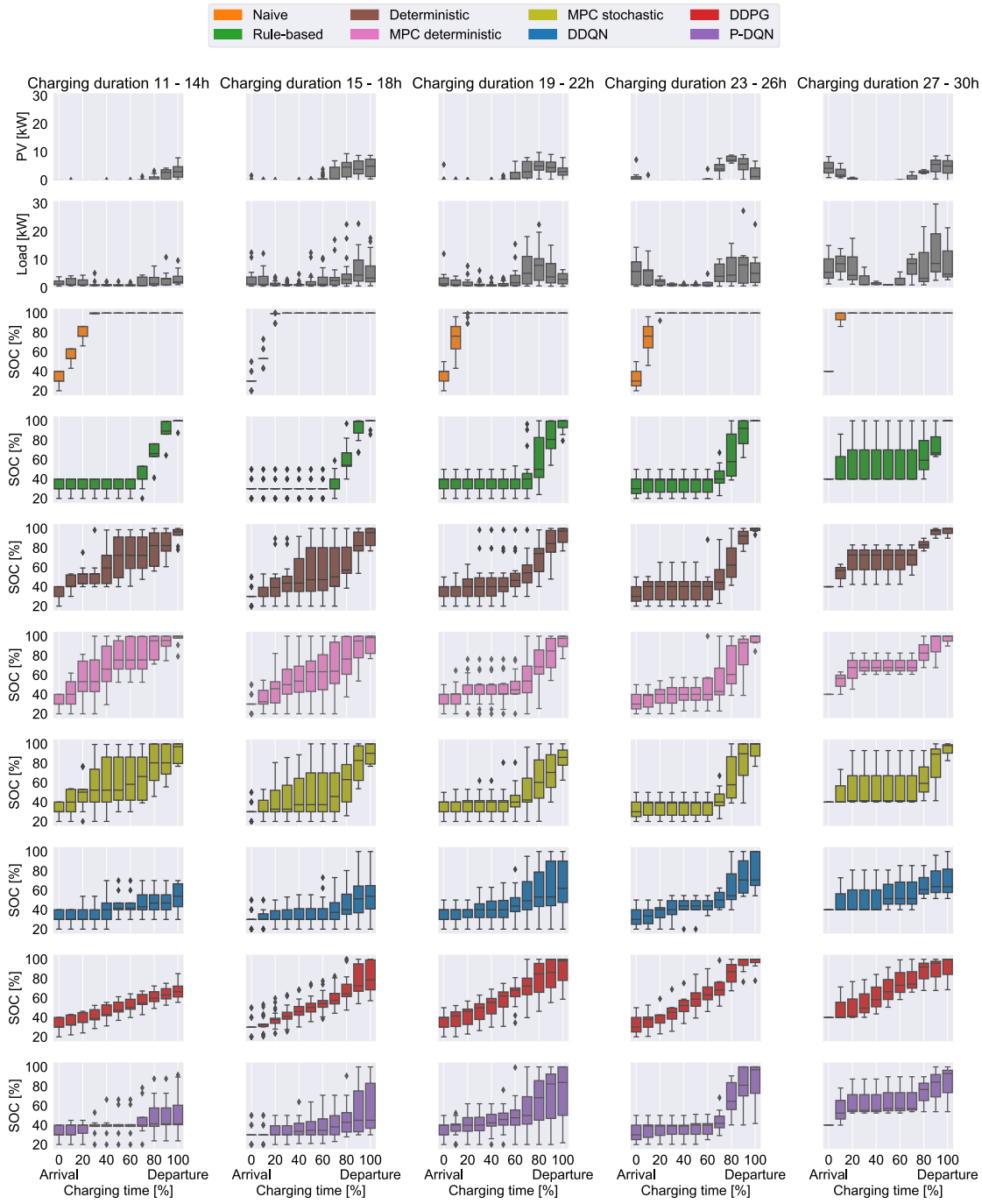



**Fig. 10.** Comparison of SOC evolution across episodes with binned charging duration.

RLC algorithms obtain less $SOC_{dep}$ satisfaction, whereas 15% of DDQN episodes and 21% of P-DQN episodes make the EV unsuited for driving purposes. Instead, the DDPG algorithm does not exhibit any episodes with low SOC satisfaction. However, the 70% share of high satisfaction episodes is lower than for non-RLC algorithms. The $SOC_{dep}$ for DDQN lies primarily in the middle-satisfaction zone, while for P-DQN in the high-satisfaction zone.

Table 5 lists the total PV self-consumption and total energy purchased from the grid for all algorithms on the whole test dataset, with a deterministic algorithm setting the baseline for comparison. The Naive algorithm confirms the need for deploying EV charging control by showcasing the lowest PV self-consumption value and the

**Table 4**
Share of episodes with various SOC at departure satisfaction levels.

| Algorithm | Low [%] | Medium [%] | High [%] |
|---|---|---|---|
| Naive | 0 | 0 | 100 |
| Rule-based | 0 | 0 | 100 |
| Deterministic | 0 | 0 | 100 |
| MPC deterministic | 0 | 0 | 100 |
| MPC stochastic | 0 | 0 | 100 |
| DDQN | 15 | 52 | 33 |
| DDPG | 0 | 30 | 70 |
| P-DQN | 21 | 32 | 47 |

**Table 5**
Total PV self-consumption and total energy purchased from the grid.

| Algorithm | PV self-consumption | Energy purchased [MWh] |
|---|---|---|
| Naive | 0.664 | 3.359 |
| Rule-based | 0.808 | 3.115 |
| Deterministic | 0.829 | 3.084 |
| MPC deterministic | 0.778 | 3.170 |
| MPC stochastic | 0.817 | 3.100 |
| DDQN | 0.762 | 2.860 |
| DDPG | 0.727 | 3.123 |
| P-DQN | 0.774 | 2.878 |

**Table 6**
The algorithms' execution time on test dataset.

| Algorithm | Time [s] |
|---|---|
| Naive | 0.21 |
| Rule-based | 0.22 |
| Deterministic | 21.04 |
| MPC deterministic | 772.48 |
| MPC stochastic | 3670.66 |
| DDQN | 2.16 |
| DDPG | 2.27 |
| P-DQN | 2.57 |

highest amount of energy purchased from the grid. Despite its non-sophisticated nature, the RBC algorithm demonstrates near-optimal performance, thus justifying its wide-spread usage in control applications nowadays. The MPC stochastic exhibits the closest to the optimal performance by leveraging the power of both PV and load demands forecasts. The MPC deterministic performs in-line with RLC approaches. The P-DQN and DDQN algorithms demonstrate the highest total PV self-consumption among RLC algorithms, while the total purchased energy is the lowest. Therefore, despite the obvious potential for higher PV self-consumption, one can hypothesize that more energy could have been purchased to increase $SOC_{dep}$. Instead, the DDPG algorithm did not realize its full potential to harvest all available PV generation.

Table 6 summarizes the execution time of the algorithms on the test dataset. The pre-training duration of the RLC algorithms is not considered as part of the execution time. Once the algorithm is trained offline, it can be efficiently utilized for decision-making. The Naive, RBC, and RLC algorithms execute within seconds, while the MPC deterministic and MPC stochastic require approximately 12 min and 1 h to execute. The latter's execution time makes it unsuitable for online applications, as the chosen time resolution of the problem is equal to 1 h. The MPC deterministic can be deployed online; however, it requires significantly more computational power than RLC algorithms.

To summarize the results described in this section, we highlight the key takeaways on the algorithms' performance on the EV charging control problem:

- The Naive, conventional charging attitude of the majority of the EV drivers nowadays leads to low PV self-consumption due to extensive charging using the grid-supplied power. Although the latter results in high EV charging costs, the Naive algorithm guarantees $SOC_{max}$ at departure, thus alleviating the range anxiety problem.
- The deterministic optimization algorithm, despite providing the optimal charging strategy, cannot be considered realistic as it assumes complete knowledge of the future PV generation and load demand values. Therefore, it can be used as a baseline to compare other algorithms' performances but should not be considered a standalone EV charging strategy.
- The RBC algorithm demonstrates near-optimal performance with simple rule formulation, low execution time, and high PV self-consumption and $SOC_{dep}$ values, thus justifying its wide-spread application for control problems nowadays. However, one should carefully consider RBC for problems with increased complexity. The addition of such features as V2G, multiple EVs, and dynamic pricing can result in difficulties in the formulation and verification of the RBC rules.
- The MPC algorithms demonstrate their ability to achieve the PV self-consumption and $SOC_{dep}$ objectives efficiently. However, one should keep in mind that MPC algorithms' formulation requires a full mathematical model of the physical system, development and integration of forecasting instruments, high computational resources, and long execution times to provide decisions. Following the increasing trend of instantaneous decision-making, MPC algorithms' utilization might not respond to the online implementation's future needs. Moreover, the growing complexity of physical systems demands more complicated mathematical models, which increase the execution times even further.
- The RLC algorithms prove their ability to be used for EV charging control despite the varying performance among considered models. With the increasing abundance of data and facilitated access to computing power, one can argue to improve the RLC algorithms' performance in the future significantly. Almost instantaneous decision-making of RLC algorithms leaves a great promise for real-time applications and poses a serious challenge to MPC algorithms with their lengthy execution times. Besides, the increasing complexity of future mobility systems can be efficiently handled by RLC through the utilization of collected big data for learning. Moreover, the model-free nature of RLC method will remain an advantage, eliminating the need for extremely complex and tedious to formulate mathematical models.

## 6. Conclusion

In this work, we have demonstrated the application of reinforcement learning to the EV charging control problem, focusing on a simple energy system composed of a utility grid, building load, PV generation, and a single EV. Particularly, we have proposed three mathematical formulations of the problem in the form of MDPs that differ by the type of action space. Moreover, we have extended the pool of EV charging control objectives by focusing on maximizing PV self-consumption and EV state of charge at departure simultaneously. To resolve the suggested MDP formulations, we have deployed the double deep Q-networks learning, deep deterministic policy gradient, and parametrized deep Q-networks learning RLC algorithms for discrete, continuous, and parametrized action spaces, respectively. Throughout a comprehensive benchmarking procedure conducted on a held-out test dataset, we have compared the RLC method with naive, RBC, deterministic optimization, and MPC deterministic and stochastic approaches. The comparison has shown that despite a slightly lesser performance of RLC algorithms on the chosen objectives, the RLC approach has exhibited its consistency in delivering applicable EV charging control strategies. Moreover, the reinforcement learning methodology has demonstrated a great potential for efficient online implementations and a natural fit to the growing complexity of future energy systems characterized by the abundance of data.

Future work to enhance the application of deep RL to the EV charging control should be conducted in four main directions:

- Additional sources of flexibility, such as heat pumps, energy storage systems, and boilers, should be included in the energy system's representation to reflect the complexity of real-world microgrids. The OpenAI Gym environments developed in this work can serve as a basis for successfully implementing these components, while the proposed algorithms can still be applicable. However, one would need to redefine the notion of an episode for better usage of additional flexibility sources.
- The EV's vehicle-to-grid capability has to be considered to utilize the EV parking time better and provide the demand–response services. The suggested parametrized implementation of the action

space would be the best fit for incorporating the vehicle-to-grid option while considering the necessary constraints on simultaneous purchasing and selling of the electricity from and to the grid.
- The suggested deep RL approach has to be extended to multiple EVs and potentially multiple hotels and PV installations. Moreover, one can choose to switch from single-agent to multi-agent methodology to explore how RL agents cooperate towards common or competitive goals.
- The search for other reinforcement learning methods that can successfully merge multiple, sometimes conflicting, objectives has to be continued. Particularly, one can analyze various reward schemes and focus on specific methodology to determine appropriate weights when fusing several objectives in continuous rewards and deciding on their prioritization. Alternatively, the application of Pareto reinforcement learning for the multi-objective EV charging problem has to be tested.

## CRediT authorship contribution statement

**Marina Dorokhova:** Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Writing – original draft, Visualization. **Yann Martinson:** Methodology, Software, Investigation, Visualization. **Christophe Ballif:** Writing - review & editing, Supervision. **Nicolas Wyrsch:** Conceptualization, Validation, Writing – review & editing, Supervision, Funding acquisition.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgments

This research is part of the activities of the Swiss Centre for Competence in Energy Research on the Future Swiss Electrical Infrastructure (SCCER-FURIES), which is financially supported by the Swiss Innovation Agency, Switzerland (Innosuisse - SCCER program).

## References

[1] European Environmental Agency. Greenhouse gas emissions from transport in europe. 2019, URL https://www.eea.europa.eu/data-and-maps/indicators/transport-emissions-of-greenhouse-gases/transport-emissions-of-greenhouse-gases-12 [Accessed September 23, 2020].
[2] International Energy Agency. Global EV outlook. 2020, URL https://www.iea.org/reports/global-ev-outlook-2020 [Accessed September 23, 2020].
[3] Hauke E, Hensley R, Knupfer S, Shivika S. The potential impact of electric vehicles on global energy systems. McKinsey Center Future Mobil 2018;1–8.
[4] Kathirgamanathan A, De Rosa M, Mangina E, Finn DP. Data-driven predictive control for unlocking building energy flexibility: A review. Renew Sustain Energy Rev 2021;135:110120. http://dx.doi.org/10.1016/j.rser.2020.110120.
[5] Vázquez-Canteli JR, Nagy Z. Reinforcement learning for demand response: A review of algorithms and modeling techniques. Appl Energy 2019;235:1072–89. http://dx.doi.org/10.1016/j.apenergy.2018.11.002.
[6] Mason K, Grijalva S. A review of reinforcement learning for autonomous building energy management. Comput Electr Eng 2019;78:300–12. http://dx.doi.org/10.1016/j.compeleceng.2019.07.019.
[7] Guan C, Wang Y, Lin X, Nazarian S, Pedram M. Reinforcement learning-based control of residential energy storage systems for electric bill minimization. In: 2015 12th annual ieee consumer communications and networking conference (ccnc). Las Vegas, USA: IEEE; 2015, p. 637–42. http://dx.doi.org/10.1109/CCNC.2015.7158054.
[8] Lee S, Choi D-H. Reinforcement learning-based energy management of smart home with rooftop solar photovoltaic system, energy storage system, and home appliances. Sensors 2019;19(18):3937. http://dx.doi.org/10.3390/s19183937.
[9] Kim S, Lim H. Reinforcement learning based energy management algorithm for smart energy buildings. Energies 2018;11(8):2010. http://dx.doi.org/10.3390/en11082010.
[10] Wan Z, He HLH, Prokhorov D. Model-free real-time EV charging scheduling based on deep reinforcement learning. IEEE Trans Smart Grid 2018;10:5246–57. http://dx.doi.org/10.1109/TSG.2018.2879572.
[11] Chiş A, Lundén J, Koivunen V. Reinforcement learning-based plug-in electric vehicle charging with forecasted price. IEEE Trans Veh Technol 2016;66(5):3674–84. http://dx.doi.org/10.1109/TVT.2016.2603536.
[12] Sadeghianpourhamami N, Deleu J, Develder C. Definition and evaluation of model-free coordination of electrical vehicle charging with reinforcement learning. IEEE Trans Smart Grid 2019;11(1):203–14. http://dx.doi.org/10.1109/TSG.2019.2920320.
[13] Lee J, Lee E, Kim J. Electric vehicle charging and discharging algorithm based on reinforcement learning with data-driven approach in dynamic pricing scheme. Energies 2020;13(8):1950. http://dx.doi.org/10.3390/en13081950.
[14] Mocanu E, Mocanu DC, Nguyen PH, Liotta A, Webber ME, Gibescu M, et al. On-line building energy optimization using deep reinforcement learning. IEEE Trans Smart Grid 2018;10(4):3698–708. http://dx.doi.org/10.1109/TSG.2018.2834219.
[15] Wu D, Rabusseau G, Francois-Lavet V, Precup D, Boulet B. Optimizing home energy management and electric vehicle charging with reinforcement learning. In *2018 16th workshop at the federated ai meeting: adaptive learning agents*, Stockholm, Sweden, 2018, p. 1–8.
[16] Zhang F, Yang Q, An D. CDDPG: A deep reinforcement learning-based approach for electric vehicle charging control. IEEE Internet Things J 2020. http://dx.doi.org/10.1109/jiot.2020.3015204.
[17] Ding T, Zeng Z, Bai J, Qin B, Yang Y, Shahidehpour M. Optimal electric vehicle charging strategy with Markov decision process and reinforcement learning technique. IEEE Trans Ind Appl 2020;9994. http://dx.doi.org/10.1109/tia.2020.2990096.
[18] Wan Z, Li H, He H. Residential energy management with deep reinforcement learning. In: 2018 international joint conference on neural networks (ijcnn). Rio de Janeiro, Brazil: IEEE; 2018, p. 1–7. http://dx.doi.org/10.1109/IJCNN.2018.8489210.
[19] Yu L, Xie W, Xie D, Zou Y, Zhang D, Sun Z, et al. Deep reinforcement learning for smart home energy management. IEEE Internet Things J 2019;7(4):2751–62. http://dx.doi.org/10.1109/JIOT.2019.2957289.
[20] Ye Y, Qiu D, Wu X, Strbac G, Ward J. Model-free real-time autonomous control for a residential multi-energy system using deep reinforcement learning. IEEE Trans Smart Grid 2020;11(4):3068–82. http://dx.doi.org/10.1109/tsg.2020.2976771.
[21] Li H, Wan Z, He H. A deep reinforcement learning based approach for home energy management system. In: 2020 ieee power & energy society innovative smart grid technologies conference (isgt). Washington, USA: IEEE; 2020, p. 1–5. http://dx.doi.org/10.1109/isgt45199.2020.9087647.
[22] Shin M, Choi DH, Kim J. Cooperative management for PV/ESS-Enabled electric vehicle charging stations: A multiagent deep reinforcement learning approach. IEEE Trans Ind Inf 2020;16:3493–503. http://dx.doi.org/10.1109/TII.2019.2944183.
[23] Fang X, Wang J, Song G, Han Y, Zhao Q, Cao Z. Multi-agent reinforcement learning approach for residential microgrid energy scheduling. Energies 2019;13:1–26. http://dx.doi.org/10.3390/en13010123.
[24] Claessens BJ, Vandael S, Ruelens F, De Craemer K, Beusen B. Peak shaving of a heterogeneous cluster of residential flexibility carriers using reinforcement learning. In: IEEE pes isgt europe 2013. Copenhagen, Denmark; 2013, p. 1–5. http://dx.doi.org/10.1109/ISGTEurope.2013.6695254.
[25] Busoniu L, Babuska R, De Schutter B. Multi-agent reinforcement learning: An overview. Stud Comput Intell 2010;310:183–221.
[26] Brockman G, Cheung V, Pettersson L, Schneider J, Schulman J, Tang J, et al. Openai gym. 2016, arXiv preprint arXiv:1606.01540.
[27] Andrychowicz M, Wolski F, Ray A, Schneider J, Fong R, Welinder P, et al. Hindsight experience replay. In: Advances in neural information processing systems. 2017, p. 5048–58.
[28] Luthander R, Widén J, Nilsson D, Palm J. Photovoltaic self-consumption in buildings: A review. Appl Energy 2015;142:80–94. http://dx.doi.org/10.1016/j.apenergy.2014.12.028.
[29] Marra F, Yang GY, Træholt C, Larsen E, Rasmussen CN, You S. Demand profile study of battery electric vehicle under different charging options. In: 2012 ieee power and energy society general meeting. IEEE; 2012, p. 1–7.
[30] Vecerik M, Hester T, Scholz J, Wang F, Pietquin O, Piot B, et al. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. 2017, arXiv preprint arXiv:1707.08817.
[31] Plappert M, Andrychowicz M, Ray A, McGrew B, Baker B, Powell G, et al. Multi-goal reinforcement learning: Challenging robotics environments and request for research. 2018, arXiv preprint arXiv:1802.09464.
[32] Van Hasselt H, Guez A, Silver D. Deep reinforcement learning with double q-learning. 2015, arXiv preprint arXiv:1509.06461.
[33] Hasselt HV. Double Q-learning. In *Advances in neural information processing systems 23 (nips)*, Vancouver, Canada, 2010, p. 2613–21.
[34] Lillicrap TP, Hunt JJ, Pritzel A, Heess N, Erez T, Tassa Y, et al. Continuous control with deep reinforcement learning. 2015, arXiv preprint arXiv:1509.02971.

[35] Uhlenbeck GE, Ornstein LS. On the theory of the Brownian motion. Phys Rev 1930;36:823–41. http://dx.doi.org/10.1103/PhysRev.36.823.
[36] Liessner R, Schmitt J, Dietermann A, Bäker B. Hyperparameter optimization for deep reinforcement learning in vehicle energy management.. In: ICAART 2019 - 11th international conference on agents and artificial intelligence. Prague, Czech Republic; 2019, p. 134–44. http://dx.doi.org/10.5220/0007364701340144.
[37] Masson W, Ranchod P, Konidaris G. Reinforcement learning with parameterized actions. 2015, arXiv preprint arXiv:1509.01644.
[38] Fan Z, Su R, Zhang W, Yu Y. Hybrid actor-critic reinforcement learning in parameterized action space. In: IJCAI international joint conference on artificial intelligence. Macao, China; 2019, p. 2279–85. http://dx.doi.org/10.24963/ijcai.2019/316.
[39] Hausknecht M, Stone P. Deep reinforcement learning in parameterized action space. 2015, arXiv preprint arXiv:1511.04143.
[40] Xiong J, Wang Q, Yang Z, Sun P, Han L, Zheng Y, et al. Parametrized deep q-networks learning: Reinforcement learning with discrete-continuous hybrid action space. 2018, arXiv preprint arXiv:1810.06394.
[41] Bester CJ, James SD, Konidaris GD. Multi-pass Q-networks for deep reinforcement learning with parameterised action spaces. 2019, arXiv preprint arXiv:1905.04388.
[42] Nair A, McGrew B, Andrychowicz M, Zaremba W, Abbeel P. Overcoming exploration in reinforcement learning with demonstrations. In: 2018 ieee international conference on robotics and automation (icra). Brisbane, Australia: IEEE; 2018, p. 6292–9. http://dx.doi.org/10.1109/ICRA.2018.8463162.
[43] Zuo S, Wang Z, Zhu X, Ou Y. Continuous reinforcement learning from human demonstrations with integrated experience replay for autonomous driving. In: 2017 ieee international conference on robotics and biomimetics (robio). Macau SAR, China: IEEE; 2017, p. 2450–5. http://dx.doi.org/10.1109/ROBIO.2017.8324787.
[44] Val d'Herens. Service green mobility. 2019, URL https://www.valdherens.ch/en/service-green-mobility-fp45415 [Accessed December 28, 2020].
[45] Innosuisse. Digitalisation in energy and mobility via SCCER. 2019, URL https://www.innosuisse.ch/inno/en/home/promotion-initiatives/impulsprogramm_digitalisierung.html [Accessed December 28, 2020].
[46] SERI. Action plan for education, research and innovation (eri) 2019–2020. Digitalization report, 2020, URL https://www.sbfi.admin.ch/sbfi/en/home/eri-policy/digitalisation.html [Accessed December 28, 2020].
[47] Electric vehicle database. Citroen C-zero. 2020, URL https://ev-database.org/car/1094/Citroen-C-Zero [Accessed October 6, 2020].
[48] Henderson P, Islam R, Bachman P, Pineau J, Precup D, Meger D. Deep reinforcement learning that matters. 2017, arXiv preprint arXiv:1709.06560.