 doi:10.20944/preprints202103.0592.v1

Article

Reinforcement Learning for Electric Vehicle Charging using Dueling Neural Networks

Gargya Gokhale[1]*, Bert Claessens[2] and Chris Develder[1]

[1] IDLab, Ghent University – imec;
[2] Eindhoven University of Technology
* Correspondence: gargya.gokhale@ugent.be

Abstract: We consider the problem of coordinating the charging of an entire fleet of electric vehicles (EV), using a model-free approach, i.e., purely data-driven reinforcement learning (RL). The objective of the RL-based control is to optimize charging actions, while fulfilling all EV charging constraints (e.g., timely completion of the charging). In particular, we focus on batch-mode learning and adopt fitted Q-iteration (FQI). A core component in FQI is approximating the Q-function using a regression technique, from which the policy is derived. Recently, a dueling neural networks architecture was proposed and shown to lead to better policy evaluation in the presence of many similar-valued actions, as applied in a computer game context. The main research contributions of the current paper are that (i) we develop a dueling neural networks approach for the setting of joint coordination of an entire EV fleet, and (ii) we evaluate its performance and compare it to an all-knowing benchmark and an FQI approach using EXTRA trees regression technique, a popular approach currently discussed in EV related works. We present a case study where RL agents are trained with an ϵ–greedy approach for different objectives, i.e., (a) cost minimization, and (b) maximization of self-consumption of local renewable energy sources. Our results indicate that RL agents achieve significant cost reductions (70–80 %) compared to a business-as-usual scenario without smart charging. Comparing the dueling neural networks regression to EXTRA trees indicates that for our case study's EV fleet parameters and training scenario, the EXTRA trees-based agents achieve higher performance in terms of both lower costs (or higher self-consumption) and stronger robustness, i.e., less variation among trained agents. This suggests that adopting dueling neural networks in this EV setting is not particularly beneficial as opposed to the Atari game context from where this idea originated.

Keywords: Electric vehicles, batch reinforcement learning, dueling neural networks, fitted Q-iteration.

1. Introduction

Over the last decade, there has been an unprecedented increase in the usage of EVs, and this trend is expected to continue over the coming decade [1]. EVs are a key element in the energy transition process, providing new opportunities as flexible load assets: they tend to be parked and connected to a charging station longer than needed to complete the charging [2]. As discussed in [3], the flexibility of EVs can be used to provide services to different stakeholders in smart grids such as ancillary services for local grid management, cost benefits for individual EV owners and improving local renewable energy consumption. In this paper we focus on an operator responsible for the charging of a fleet of parked EVs. The operator ensures that each EV is charged to a user-defined energy level and often performs this by charging each EV as soon as it arrives. We will consider two main scenarios for the operator to deviate from this strategy: (i) minimize the total cost of charging, and (ii) maximize the consumption of locally generated renewable energy (PV). The charging strategies for both these scenarios are constrained by the need to complete the charging of each EV before its departure.

An established method for such control problems is Model Predictive Control (MPC) [3], which uses a model of the system and an optimisation algorithm to solve a receding horizon problem [4]. Although MPC achieves state-of-the-art, it relies on an accurate model of the system that can be expensive to obtain and susceptible to real-world uncertainties. With recent advancements in data-driven control, Reinforcement Learning (RL) techniques have made a strong case for their application pertaining to such problems where a system model is expensive or difficult to obtain [5]. RL relies on the interactions between an agent (controller) and an environment, and learn to take optimal control decisions by learning the dynamics of this environment [6]. Thus, it is excellently suited for the EV fleet charging problem, where dynamics are often dependent on uncertain arrival, departure times and energy requirements of individual EVs.

As discussed further in Section 1.1, several RL based approaches have been researched previously for DR related control problems and each approach has its own merit. Given the specifics of this problem—a continuous state representation and a set of discrete actions—we focus on the batch reinforcement learning technique of fitted Q-iteration (FQI) [7]. As discussed in [8], FQI exploits data more efficiently compared to other RL algorithms such as policy optimization or deep Q-networks. It purely relies on past transitions between the agent and the environment, and trains a regression model to approximate the Q-function and derive optimum control decisions. With this, we aim to determine optimum charging actions for the fleet of EVs without any explicit knowledge about individual EV parameters like arrival, departure times or energy requirements. Several regression techniques have been studied previously in RL literature, including tree based methods and deep learning based approaches. Amongst the latter, a novel technique of dueling neural networks has been shown to significantly improve the performance of RL agents trained to play Atari games [9]. Based on this, we aim to investigate the impact of dueling neural networks, towards the performance of fitted Q-iteration in the above mentioned EV charging problem.

The main contributions of this paper can be summarized as follows:

1. To design a dueling neural networks architecture as a functional approximator in fitted Q-iteration and implement it in the EV fleet problem to determine optimum charging actions (Sections 2–3).
2. To evaluate and compare the performance of agents using this dueling neural networks architecture with EXTRA trees, a popular regression technique for FQI. Different objectives and information settings will be used to compare both these methods. (Sections 4–5).

1.1. Related Works

Several works have used reinforcement learning techniques in the context of demand response and electric vehicle charging [3]. Here, we provide an overview of work related to different RL techniques and their application to EV charging. The Q-learning algorithm has been extensively researched for its application in demand response (DR) [10,11]. However, the algorithm requires discrete state and action spaces, generally represented as a Q-table and updated after every transition. This is a major drawback of this algorithm leading to inefficient use of data and a long time for convergence.

Following recent advances in computational capacities and deep learning techniques, the current state-of-the-art focuses on approximating the Q-function using functional approximators (regression models) including deep neural networks. These approaches can be classified into two main categories: (i) policy optimization and, (ii) value iteration. For (i), the work presented in [12] uses a constrained policy optimisation algorithm to train an RL agent for an EV charging/discharging problem. This is a policy search based approach where a deep neural network is trained to take actions that optimize the cost of electricity consumption of an EV fleet. Additionally, a constrained

 doi:10.20944/preprints202103.0592.v1

MDP formulation is used to account for the randomness in arrival, departure times of EVs, following which the policy optimization based RL agent learns an optimum charging schedule. The experiments were performed for a single EV with training and test data equivalent to a year's data each. The results are presented for a single EV and indicate significantly better performance of the proposed approach as compared to other commonly used methods such as Deep Q Networks and Actor critic approaches. In contrast, we present a method of coordinating the charging of multiple EVs with significantly less training data (~100 days).

For (ii), a value iteration approach based on the DQN algorithm [13] has been studied in [14]. Here, Wan *et al.* implement the DQN algorithm for training an RL agent to take optimum charging and discharging decisions for a single EV. A deep neural network is used that forecasts future electricity prices and then approximates the Q-function to take the best actions. While [14] studies the charging of a single EV, Lee *et al.* investigate the use of a DQN based RL agent for the charging and discharging of a fleet of residential EVs [15]. The training is carried out on a training data of 25000 days obtained using a kernel density estimation of real world data corresponding to 25 days. Following the training, a cost reduction of about 10–20% was reported. To the contrary, our method uses less training data (100 days) to obtain an optimum charging policy for the fleet of EVs. Aside from DQN, another approach using batch RL is FQI. Vandael *et al.* apply FQI to a similar setting as ours, i.e., they jointly coordinate the charging of multiple EVs [16]. An EXTRA trees [17] based RL agent is used to determine a day ahead consumption plan for a fleet of EVs. The agent is trained in about 30 days and achieves results approximately equal to a stochastic programming based benchmark. Similar to this, we propose a dueling neural networks architecture instead of the EXTRA trees regression technique to train the agent and obtain optimum control policies.

The articles mentioned above, specifically [12,15] present two methods that produce significantly better results compared to the other works presented but also require a large training dataset. Conversely, using FQI as presented in [16] proves to be a data efficient. Additionally, with the specific problem definition involving a continuous state space, discrete finite action space, we focus on FQI as the RL technique to be used. Similar to the choice of algorithm, several methods have been proposed for modelling of EVs and EV fleets: a three-step modeling approach using a priority based dispatch [18]; a binning algorithm for 2D grid representation of aggregate state [19]; a constrained MDP formulation of EV charging/discharging [12]. We will adapt the three-step modeling approach as presented in [18] to model an EV fleet.

The main aim of this paper is to investigate dueling neural networks as a regression technique in FQI and assess its performance in obtaining optimal control decision in the RL setting. Based on literature, EXTRA trees regression technique is a popular choice for such problems, i.e., FQI for DR scenarios [20]. It has been shown to outperform alternative approaches such as multi-layer perceptrons, XGBBoost, bag of ELMs (extreme learning machines), etc. in the context of residential heating loads for DR [20]. Hence we will benchmark dueling neural networks against the EXTRA trees regression technique.

2. Problem Formulation

A reinforcement learning approach relates to the process of an agent learning a control policy h through observed interactions with the environment to be controlled. The decision making process of an RL agent is formulated using a Markov Decision Process (MDP) [21], as shown in Fig. 1. In this paper, we will consider discrete MDPs, where the agent interacts with the environment in discrete time steps t. An MDP is defined by its state space X, action space U, and transition function $f : X \times U \times W \to X$,

$$x_{t+1} = f(x_t, u_t, w_t). \tag{1}$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 March 2021 doi:10.20944/preprints202103.0592.v1

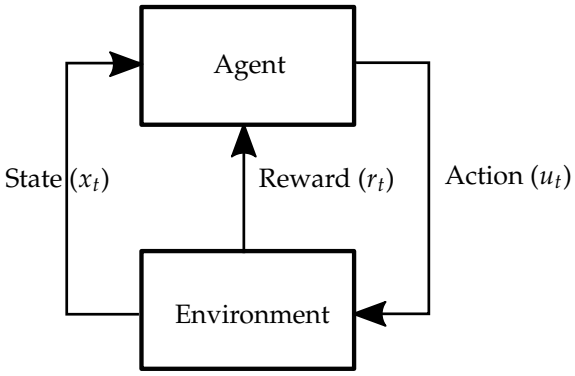

Figure 1. This figure shows a schematic representation of a general MDP. The agent influences the environment by taking action u_t, based on a state signal x_t and gets an immediate reward r_t.

The expression in Eq. (1) represents the dynamics of the environment from a state $x_t \in X$ to x_{t+1}, under action $u_t \in U$ and a random process $w_t \in W$ with a probability distribution $p_w(\cdot, x_t)$. Each state transition is accompanied by an immediate reward signal r_t, defined by a reward function $\rho : X \times U \times W \to \mathbb{R}$:

$$r_t = \rho(x_t, u_t, w_t). \tag{2}$$

The goal of RL agent is to find a policy $h : X \to U$, $u = h(x)$, that minimizes the total reward during a finite time horizon T, starting from an initial state x_1. This T-time horizon reward is denoted by $J^h(x_1)$ and is given by:

$$J^h(x_1) = \mathbb{E}\Big[\sum_{t=1}^{T} \rho(x_t, h(x_t), w_t)\Big]. \tag{3}$$

J^h can be expressed in a recursive and convenient way by using the state-action value function or Q-function [6] defined as:

$$Q^h(x_t, u_t) = \mathbb{E}\Big[\rho(x_t, u_t, w_t) + J^h(x_{t+1})\Big]. \tag{4}$$

The Q-function defined in Eq. (4) gives the expected cumulative reward over the time horizon T and hence can be used to characterize a policy h. The optimal Q-function corresponds to the best Q-function that can be obtained over all policies h and is given by:

$$Q^*(x, u) = \min_h Q^h(x, u). \tag{5}$$

The policy that gives the optimal Q-function for all state-action pairs is termed as an optimal policy and can be calculated as a greedy policy:

$$h^*(x) \in \operatorname*{argmin}_{u \in U} Q^*(x, u). \tag{6}$$

Further, the Bellman optimality equation [21] can be used to obtain the optimal Q-function as shown in Eq. (7).

$$Q^*(x, u) = \mathop{\mathbb{E}}_{w \sim p_w(\cdot, x)}\Big[\rho(x, u, w) + \min_{u' \in U} Q^*(f(x, u, w), u')\Big] \tag{7}$$

2.1. State Space Description

The state space X represents the set of all possible states of the environment, which in this case is the fleet of EVs to be charged. The charging of each EV i is defined by a set of parameters: arrival time (t_i^{arv}), departure time (t_i^{dep}), energy required for full charge

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 March 2021 doi:10.20944/preprints202103.0592.v1

(E_i^{req}), minimum required state of energy at departure (E_i^{sat}), maximum charging power ($P_i^{\max}$) and the current state of energy($x_{i,t}^{\text{phys}}$). The tuple $\Omega_{i,t} = (t_i^{\text{arv}}, t_i^{\text{dep}}, E_i^{\text{req}}, x_{i,t}^{\text{phys}}, P_i^{\max})$, constitutes the internal state of EV i and completely defines its behavior. With prior knowledge of $\Omega_{i,t}$, EV i can be controlled in a deterministic manner for any time t. Hence the internal state of the fleet ($x_t^{\text{int}} \in X$) is defined as

$$x_t^{\text{int}} = \{\Omega_{1,t}, \Omega_{2,t}, \ldots, \Omega_{N_t^{\text{con}},t}\} \tag{8}$$

where N_t^{con} represents the number of EVs connected at time t. Using this definition, the minimum and maximum energy levels for each EV i at a given time t are defined as

$$\begin{aligned} E_{i,t}^{\min} &= \max(E_i^{\text{sat}} - P_i^{\max} \cdot (t_i^{\text{dep}} - t),\ 0), \\ E_{i,t}^{\max} &= \min(E_i^{\text{req}},\ P_i^{\max} \cdot (t - t_i^{\text{arv}})). \end{aligned} \tag{9}$$

Further, the state of energy of EV i at time t is represented by $x_{i,t}^{\text{phys}}$ and satisfies the constraints given in Eq. (10).

$$E_{i,t}^{\min} \leq x_{i,t}^{\text{phys}} \leq E_{i,t}^{\max} \qquad \forall t \tag{10}$$

2.2. Action Space Description

The action space U corresponds to a set of actions that the agent can take to influence the state of the environment. An action thus corresponds to the charging power that is given to each EV i at time t and is assumed to take discrete values, 0 or $P_i^{\max}$. The state of energy of each EV is bound by Eq. (10) and hence each EV can overrule an action taken by the agent to satisfy these constraints. This is modelled using the function B, that maps the action ($u_{i,t}$) determined by the agent for EV i, to a physical control action $u_{i,t}^{\text{phys}}$ depending on the state of energy $x_{i,t}^{\text{phys}}$,

$$u_{i,t}^{\text{phys}} = B(x_{i,t}^{\text{phys}}, u_{i,t}, E_{i,t}^{\text{lim}}). \tag{11}$$

Here, $E_{i,t}^{\text{lim}}$ consists of minimum and maximum energy boundaries as defined in Eq. (9) and $B(\cdot)$ is defined in Eq. (13) as shown in Section 2.3.

2.3. Transition Function Description

The transition function f describes the transition of the environment from state x_t to x_{t+1} due to an action $u_{i,t}$ under an uncertainty w_t. However, for this problem the uncertainty (w_t) represents the uncertainty in arrival and departure times of each EV and is modelled implicitly in this formulation. Further, the transition function defines the charging process of each EV. Using a linear model and constraints defined in Eq. (10), for an EV i, the new energy state ($x_{i,t+1}^{\text{phys}}$) is modelled as:

$$x_{i,t+1}^{\text{phys}} = \begin{cases} E_{i,t}^{\min} & : (x_{i,t}^{\text{phys}} + u_{i,t}\Delta t) < E_{i,t}^{\min} \\ x_{i,t}^{\text{phys}} + u_{i,t}\Delta_t & : E_{i,t}^{\min} \leq (x_{i,t}^{\text{phys}} + u_{i,t}\Delta t) \leq E_{i,t}^{\max} \\ E_{i,t}^{\max} & : E_{i,t}^{\max} < (x_{i,t}^{\text{phys}} + u_{i,t}\Delta t) \end{cases}. \tag{12}$$

The expression in Eq. (12) represents the transition over the duration of a time slot. The actual power ($u_{i,t}^{\text{phys}}$) used by EV i for time t depends on the new state of energy and is given by:

$$u_{i,t}^{\text{phys}} = B(x_{i,t}^{\text{phys}}, u_{i,t}, E_i^{\text{lim}}) = \frac{x_{i,t+1}^{\text{phys}} - x_{i,t}^{\text{phys}}}{\Delta t}. \tag{13}$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 March 2021 doi:10.20944/preprints202103.0592.v1

2.4. Reward Description

The reward function (ρ) is modelled following the objective of the agent. In this paper, two different agent objectives are studied. Consequently, two different reward functions are used and are described as follows:

1. *Minimizing Cost*: The objective of the agent is to minimize the cost of energy consumed during the charging process. To achieve this objective, the reward signal is based on the day ahead price and corresponds to the cost of energy consumed in each time slot. The reward function ρ is given by:

$$\rho(u_{1,t}^{\text{phys}}, \ldots, u_{N_t^{\text{con}},t}^{\text{phys}}, \lambda_t) = \sum_{i=1}^{N_t^{\text{con}}} \lambda_t u_{i,t}^{\text{phys}} \Delta t, \tag{14}$$

where, N_t^{con} represents the number of EVs connected at time t and λ_t represents the day-ahead price for time t.

2. *Maximizing Self-Consumption*: The objective of the agent is to maximize self-consumption, considering locally generated solar energy. To achieve this objective, the hourly price signal (λ_t) and solar generation (P_t^{PV}) is considered. The reward is calculated as the cost of energy consumed or delivered for each time slot and is modelled as:

$$\begin{aligned} \rho(u_{1,t}^{\text{phys}}, .., u_{N_t^{\text{con}},t}^{\text{phys}}, \lambda_t, \lambda_t^{\text{PV}}, P_t^{\text{PV}}) &= \begin{cases} \lambda_t P_t^{\text{net}} & : 0 \leq P_t^{\text{net}} \\ \lambda_t^{\text{PV}} P_t^{\text{net}} & : P_t^{\text{net}} < 0 \end{cases} \\ \text{where } P_t^{\text{net}} &= \sum_{i=1}^{N_t^{\text{con}}} u_{i,t}^{\text{phys}} - P_t^{\text{PV}}, \end{aligned} \tag{15}$$

and N_t^{con} represents the number of EVs connected at time t.

3. Implementation

In this section, the implementation of our RL agent is discussed. We adopt a three-step modelling and dispatch approach previously presented in [18]. In step 1, state information of each EV is collected and represented as an aggregate. Step 2 involves using this aggregate state information to calculate the charging power for the entire fleet, followed by step 3, where this power is distributed between individual EVs following a priority based dispatch. This modelling approach provides a high-fidelity model, which is linear, scalable and substantially complex to train and test the agent.

3.1. Step 1: Aggregation

The internal state defined in Eq. (8) gives a raw representation of the state of each EV in the fleet. This representation is high-dimensional, and the dimensions vary with time depending on the number of EVs present. To avoid using this representation, alternative features were engineered. To capture time dependence, a time component (x_t^{time}) corresponding to the hour of the day was used such that,

$$x_t^{\text{time}} \in \{1, 2, \ldots, T-1, T\}.$$

Further, an aggregate state of energy of fleet was used to express a time dependent controllable state component. The aggregate energy state x_t^{agg} is given by:

$$x_t^{\text{agg}} = \sum_{i=1}^{N_t^{\text{con}}} x_{i,t}^{\text{phys}} \tag{16}$$

This aggregate representation of state of energy of fleet leads to a partially observable MDP and $\mathbf{o}_t = (x_t^{\text{time}}, x_t^{\text{agg}})$ represents an observation of the actual state of the environ-

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 March 2021 doi:10.20944/preprints202103.0592.v1

ment. This partial observability is addressed by using an approximate state [6] defined as:

$$\begin{aligned} \mathbf{x_t} &\doteq (x_t^{\text{time}}, x_{t-k}^{\text{agg}}, x_{t-(k-1)}^{\text{agg}}, \ldots, x_t^{\text{agg}}), \\ \text{where} \quad x_{t-m}^{\text{agg}} &= x_0^{\text{agg}} \qquad \forall\, m \ \text{ s.t. } \quad t-m<0. \end{aligned} \tag{17}$$

Here, k represents the number of past observations that are included in the approximate state and will be referred to as "*depth*". This is a hyperparameter that determines the quality of state information being passed on to the agent. This depth will vary for different information settings and following a sensitivity analysis, for this problem, we use depth $= 2$. Similar to the aggregate state, an aggregate agent action is obtained by aggregating over all connected EVs,

$$u_t = \sum_{i=1}^{N_t^{\text{con}}} u_{i,t} \tag{18}$$

Consequently, since we consider a charging rate of 3kW and a maximum of 100 EVs, the action space U is modified as

$$U = \{0, 30, 60, \ldots, 270, 300\}, \tag{19}$$

where all values represent aggregate charging power in kW.

3.2. Step 2: Batch Reinforcement Learning

In the second step, a control action for the entire fleet is determined. This action is selected from the action space based on Eq. (6). In this paper, Fitted Q-iteration (FQI) is used to obtain an approximation of the optimal Q-function (Q^*).

3.2.1. Fitted Q-iteration

Based on the work presented in [7], FQI uses a regression model and a batch of previous interactions between the agent and the environment to estimate an approximate optimal Q-function, $\widehat{Q}^*$ over all state-action pairs. While different regression models can be used, this paper focuses on the application of dueling neural networks. With a finite time horizon, it is hypothesised that the dueling neural networks architecture can be effective in representing the Q-function by capturing the variations caused due to the value function as well as the action advantages. Further, we will compare the dueling neural networks regression to a more conventional approach of using EXTRA trees [22], [16]. As described in Algorithm 1, FQI uses a set $\mathcal{F}$, consisting of tuples of the form $\{(\mathbf{x}_l, u_l, \mathbf{x}'_l, r_l)\}$, to calculate $\widehat{Q}^*$. Here, $\mathbf{x}_l$ is the approximate state of the environment as defined in Eq. (17) and u_l represents the action taken during this instance. Consequently, $\mathbf{x}'_l$ denotes the state reached after transition and r_l is the reward obtained. Deviating from the algorithm presented in [7], we use an ensemble of Q-functions, each corresponding to a time slot. For the final time slot, the Q-function model ($\widehat{Q}_T$) is trained on the reward values directly ($\because \widehat{Q}_{T+1} = 0 \ \ \forall (x,u) \in X \times U$). This approach ensures a better convergence by avoiding the problem of training regression models on non-stationary Q-function targets. The batch of past transitions ($\mathcal{F}$) is built during the training process following an ϵ-greedy exploration technique [6]. In this exploration technique, the agent selects an action u_k according to Eq. (20), based on an exploration probability ϵ. During the training, the exploration probability (ϵ) is initialized to 1 and reduced over the training period to obtain an optimal policy. The

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 March 2021 doi:10.20944/preprints202103.0592.v1

Algorithm 1: Fitted Q-iteration (Backward Induction)

Input: Set of four tuples $\mathcal{F} = \{(\mathbf{x}_l, u_l, \mathbf{x}'_l, r_l)\}_{l=1}^{\#\mathcal{F}}$, regression algorithm,
Initialise: $\widehat{Q} = [\widehat{Q}_1, \ldots, \widehat{Q}_T]$ on $X \times U$
for $k = T, \ldots, 2, 1$ **do**
    Build training set $\mathcal{T}$=$\{(i_l, o_l), l = 1, 2, \ldots, \#\mathcal{F}\}$:
    $i_l = (\mathbf{x}_l, u_l)$
    $r_l = \rho(x_l, u_l)$
    $o_l = r_l + \min_u \widehat{Q}_{k+1}(x'_l, u)$
    Use regression algorithm on $\mathcal{T}$ to obtain $\widehat{Q}_k$
end for
Output: $\widehat{Q}^* = [\widehat{Q}_1, \ldots, \widehat{Q}_T]$

value of ϵ is a hyperparameter and is used to strike a balance between exploration and exploitation by the agent.

$$u_k = \begin{cases} \tilde{u} \in \text{argmin}_u \, \widehat{Q}_k(x_k, u) & : \text{with probability } 1 - \epsilon \\ \text{uniform random action in } U & : \text{with probability } \epsilon \end{cases} \tag{20}$$

The training process starts with an empty set $\mathcal{F}$ and four tuples $(\mathbf{x}_l, u_l, \mathbf{x}'_l, r_l)$ are collected with each interaction between the agent and the environment, following the ϵ-greedy exploration technique mentioned previously.

3.2.2. Dueling Neural Network

A dueling neural networks architecture explicitly separates the representation of state values and state-dependent action advantages [9]. The *value function* is based on the long-term reward obtained by starting from a particular state. Contrary to this, the *advantage function* captures the relative importance of an action in the current state. With this representation, dueling neural networks can hence capture effectively the variations in Q-function that are caused due to immediate dispatch actions and due to long term charging state values. In [9], it has been shown that agents using dueling neural networks architectures perform better than vanilla deep neural networks for the Atari game environments. Hence, we investigate if agents trained using a dueling neural networks architecture would also be more effective for our EV fleet problem. Figure 2 shows the schematic representation of a dueling neural networks architecture.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 March 2021 doi:10.20944/preprints202103.0592.v1

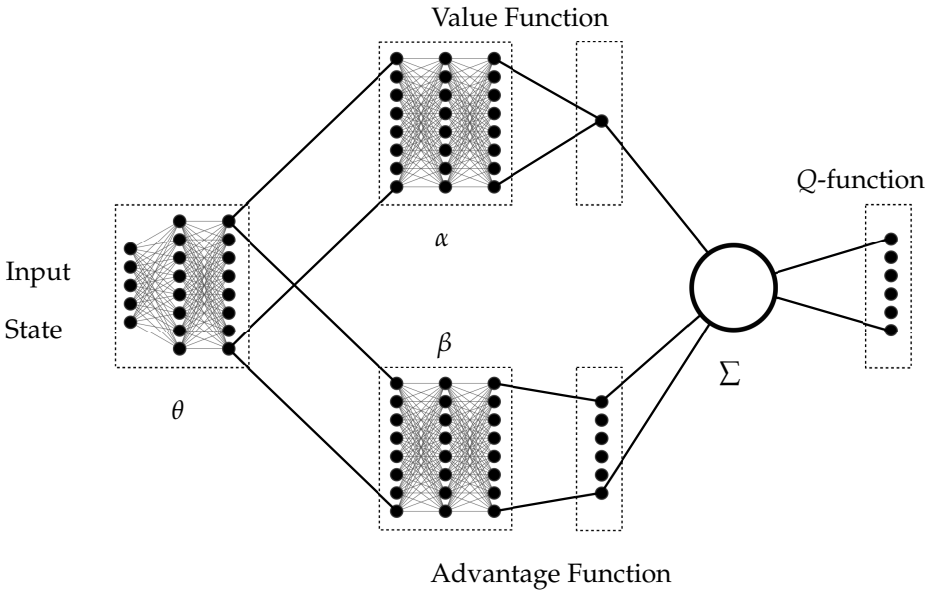

Figure 2. Schematic representation of dueling neural networks architecture. The input state signal is given to a common input module of fully connected layers represented by parameters θ. The output of this module is then split into two streams for the value and advantage function approximations, with the flow represented by the black lines. The value function stream represented by α, gives a single output corresponding to the value of input state. The advantage function stream represented by β, gives the relative advantages corresponding to all possible actions. The aggregation of these streams is represented by $\sum$ and gives the Q-function approximation for the input state signal corresponding to all actions in U using Eq. (23).

As discussed previously, the advantage function measures the relative importance of taking an action u_t, in a particular state x_t and is defined in Eq. (21), where V^h represents the value function following policy h.

$$A^h(x_t, u_t) = Q^h(x_t, u_t) - V^h(x_t) \tag{21}$$

For the optimal case, if * represents the optimal policy and u^* represents the best action in state x, then the advantage and value functions are given by Eq. (22).

$$\begin{aligned} V^*(x) &= Q^*(x, u^*), \\ A^*(x, u^*) &= 0, \end{aligned} \tag{22}$$

where Q^*, V^*andA^* are the optimal Q, value and action functions respectively. From Eq. (21), it is clear that given the value of Q-function for a state-action pair, it is not possible to uniquely recover the value function and the advantage function values. This issue is addressed by forcing the advantage function estimator to zero for the best action based on Eq. (22), as discussed in [9]. This aggregation is represented as "$\sum$" in Fig. 2 and gives the Q-function estimate for the given input state and all possible actions in U. As shown in Fig. 2, the input module, value function stream and advantage function stream are represented by learnable parameters θ, α, β respectively. The value function and advantage function approximations are expresssed in the parametric form as $V(x;\theta,\alpha)$ and $A(x,u;\theta,\beta)$, respectively. The parameterized Q-function approximation $Q(x,u;\theta,\alpha,\beta)$ is then obtained using Eq. (23).

$$\begin{aligned} Q(x,u;\theta,\alpha,\beta) = V(x;\theta,\alpha) + \\ \big(A(x,u;\theta,\beta) - \min_{u'} A(x,u';\theta,\beta)\big). \end{aligned} \tag{23}$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 March 2021 doi:10.20944/preprints202103.0592.v1

3.3. Step 3: Real Time Control

In the third step, a collective charging action u_t is selected as a greedy policy Eq. (6) using the learned Q-function from Algorithm 1. The collective charging power u_t is then distributed between all EVs following a priority-based dispatch algorithm [18]. In this dispatch algorithm, for each time slot all EVs are assigned a corner priority dependant on their departure times, energy requirements and charging capacities. Following this, EVs with a higher corner priority are given preference to charging, while the charging is delayed for EVs with lower corner priorities. This priority dispatch algorithm is modelled using Eq. (24)–(27). The power demand of an individual EV i at time t is expressed as $f_{i,t}(p)$, which is a function of priority p:

$$f_{i,t}(p) = \begin{cases} P_i^{\max} & : 0 \leq p \leq p_{i,t}^c \\ 0 & : p > p_{i,t}^c \end{cases}, \tag{24}$$

where $p_{i,t}^c$ is the corner priority for EV i at time t and defined as:

$$p_{i,t}^c = \frac{E_i^{\text{req}} - x_{i,t}^{\text{phys}}}{P_i^{\max} \cdot (t_i^{\text{dep}} - t)}. \tag{25}$$

It is a heuristic that indicates the priority assigned to an EV for charging. The closer the corner priority is to 0, the lower are the chances of the EV being dispatched (charged). At the fleet level, the power demand of all EVs is aggregated:

$$P_t^{\text{dem}}(p) = \sum_{i=1}^{N_{\text{ev}}} f_{i,t}(p). \tag{26}$$

The collective charging action u_t is distributed between individual EVs following a priority dispatch technique, by first calculating an equilibrium priority:

$$p_t^{\text{eq}} = \underset{p}{\text{argmin}} \, |P_t^{\text{dem}}(p) - u_t|. \tag{27}$$

This equilibrium priority is then passed on to all EVs. The EVs consume according to $u_{i,t} = f_{i,t}(p_t^{\text{eq}})$. The corner priority heuristic allows the agent to learn the charging behavior of EVs and influence their charging decisions. Likewise, EVs can override the agent to satisfy individual energy constraints given by Eq. (9).

4. Simulation Methodology

This section describes the methodology of simulations performed. First, we present the environment and regression model parameters. Then, we outline the simulation setup for each objective. These simulations are based on an ϵ-greedy exploration based implementation of FQI (Algorithm 1) for different objectives of the agent as described in Section 2.4.

4.1. EV Fleet Simulation Environment

The EV fleet model described in the previous section is adopted for 100 EVs. The assumed parameters in this case are given in Table 1. The energy required is calculated based on the distance travelled by each EV (Table 1) and the energy usage per km of 0.174 kWh/km as referred from the Nissan leaf EV data [23]. Further, E_i^{sat} used in Eq. (9) is set at 75 % of energy required for full charge for each EV. The arrival and departure times are assumed to be randomly distributed in a truncated normal distribution, while a uniform random distribution is assumed for the distance travelled by each EV. The time horizon is fixed between 5 am and 8 pm and divided into 16 discrete time slots of 1 hour each.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 March 2021 doi:10.20944/preprints202103.0592.v1

Table 1: Individual EV Parameters

Arrival	$5\text{am} \leq \mathcal{N}(8\text{am}, 2\text{h}) \leq 12\text{pm}$
Departure	$3\text{pm} \leq \mathcal{N}(5\text{pm}, 2\text{h}) \leq 8\text{pm}$
Distance	$\text{Uniform}(40\text{km}, 172\text{km})$

Table 2: Model parameters

EXTRA Trees	
Parameter	Value
Number of Outputs	1
Number of Trees	50
Criterion	Mean Squared Error
Minimum sample split	3
Dueling Neural Network	
Parameter	Value
Number of Outputs	11
Activation Function	Hyperbolic Tangent
Loss Function	Mean Squared Error
Optimiser	Adam
Dropout	0.1
Number of Neurons per layer	48
Initial Module	3 layers
Value Function Stream	4 layers
Advantage Function Stream	4 layers

4.2. Regression Models

The simulations are performed using FQI with two regression models: dueling neural networks and EXTRA trees. The EXTRA trees implementation is based on the work presented in [17] and implemented using the sklearn package [24]. The parameters used for both models after hyperparameter tuning are given in Table 2.

The performance of each model needs to be compared with a business-as-usual case of charging EVs as soon as they arrive and a benchmark case where a controller takes optimum charging decisions based on perfect knowledge about the environment (i.e., EV parameters Ω). Additionally, both these models are compared to each other to investigate which model calculates the best policy. For this comparison, we define a score metric,

$$score = \frac{\text{Total Reward w/o controller} - \text{Total Reward Model}}{\text{Total Reward w/o controller} - \text{Benchmark Value}}, \tag{28}$$

where 'Total Reward w/o controller' represents a business-as-usual case where EVs charge on arrival without any control logic and 'Benchmark Value' represents the optimal solution calculated based on perfect information (Appendix G). This score metric quantifies the performance of each regression model into a numeric quantity between 0 and 1, with 1 being the best performance.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 March 2021 doi:10.20944/preprints202103.0592.v1

4.3. Simulation Description

The simulations include training 20 agents of each type (i.e., using one of the two regression models) for both objectives as described in Section 2.4. For each objective, the training span is set for 100 training days, with each training day accompanied by a new set of EV parameters (Ω). An agent follows an ϵ-greedy exploration strategy over the training period and is trained after every 10 days,[1,2] according to Algorithm 1.[3] To evaluate the agent's ability to learn EV charging patterns and user behavior, 3 validation days are used. Each validation day comprises of a new set of EV parameters (Ω) and price and PV power profiles. The profiles corresponding to July 26, September 6 and October 21, 2020 were chosen and are shown in Fig. 3 [25]. The scores obtained for each validation day are used for comparing the performances of both regression techniques. The results in Section 5, present the comparison between agents trained using two regression techniques: dueling neural networks and EXTRA Trees. However, similar simulations were performed on agents using a vanilla deep neural network. The results for these simulations were much worse than the two above-mentioned regression techniques and are presented in Appendix H.

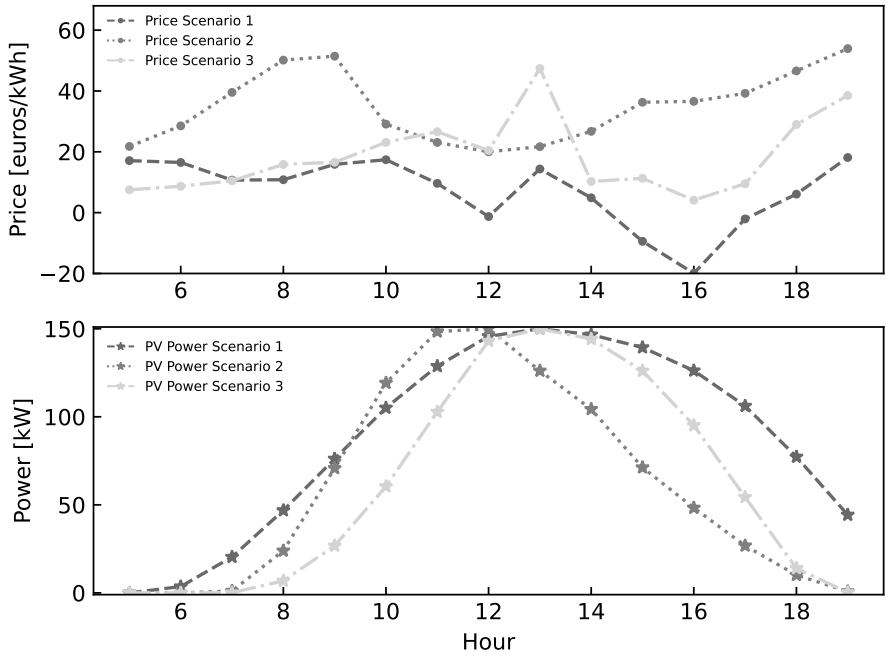

Figure 3. All Validation Profiles

5. Results and Discussions

The results for the experimental setup described in the previous section are presented and examined in this section. First, the results for the objective of Minimizing Cost are presented and later, the results for Maximizing Self-Consumption are shown. Both sets of results are focused on assessing the performance of the dueling neural networks architecture and comparing it with EXTRA trees.

[1] In practice the agent should be trained at the end of each day based on the forecasted price of the next day. To reduce the computational burden of the entire simulation, each agent was trained after every 10 days.

[2] The simulations were carried out on a Laptop with the following specifications: i7 10th Gen processor, 16 GB RAM

[3] In each simulation, 5 agents are trained simultaneously using the PyTorch Multiprocessing functionality. The total time taken for training 20 agents is about 4 hours

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 March 2021 doi:10.20944/preprints202103.0592.v1

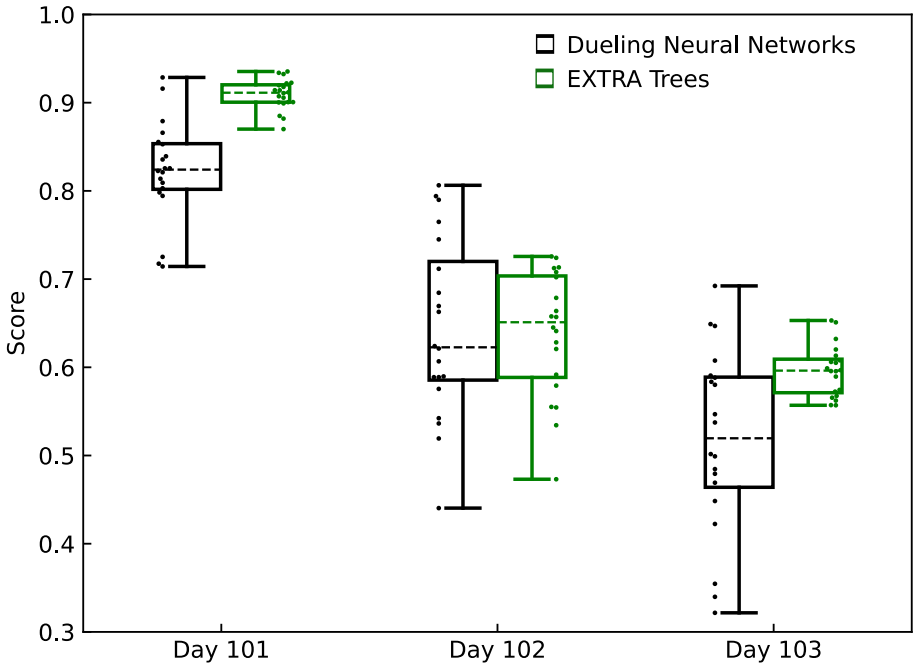

Figure 4. Comparison of performance of all agents for the 3 validation scenarios. The box plot represents the boundary of first and third quartiles, while the dashed line represents the median of the scores corresponding to the 20 agents. The dots depict the performances of individual agents.

5.1. Objective 1: Minimizing Cost

The agent objective is to minimize the cost of energy consumed during a day. As defined in Section 2.4, the agent is exposed to day ahead prices and is expected to take optimal actions depending on the variation in prices over the day. The training period includes a total of 10 different price profiles. The performance of both regression models was tested on 3 validation days. Figure 4 shows the comparison for all 20 agents of both agent classes for the 3 validation days. It can be observed in Fig. 4 that the performance of EXTRA trees based agents is slightly better than the dueling neural networks based agents. For all three validation scenarios, the median values of EXTRA trees agent are higher implying an overall better performance. Further, for days 101 and 103, the EXTRA score distribution exhibits low variance signifying a stable performance. However, as shown in Table 3, comparing maximum (best performing) scores for each validation day, the dueling neural networks based agents are better for both days 102 and 103.

Table 3: Minimizing Cost- Performance of best performing agents for all validation scenarios

	Dueling Neural Networks	EXTRA Trees
Day 101	0.93	0.94
Day 102	0.81	0.73
Day 103	0.69	0.65

The results from Table 3 suggest that both dueling neural networks and EXTRA trees based agents are capable of computing control actions that lead to significant reduction in daily operational cost (shown in Table 1). It can observed that an average cost reduction of 60% can be achieved using the best performing agents as compared to a business-as-usual case of charging EVs as soon as they arrive. Further, Fig. 5 shows the

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 March 2021 doi:10.20944/preprints202103.0592.v1

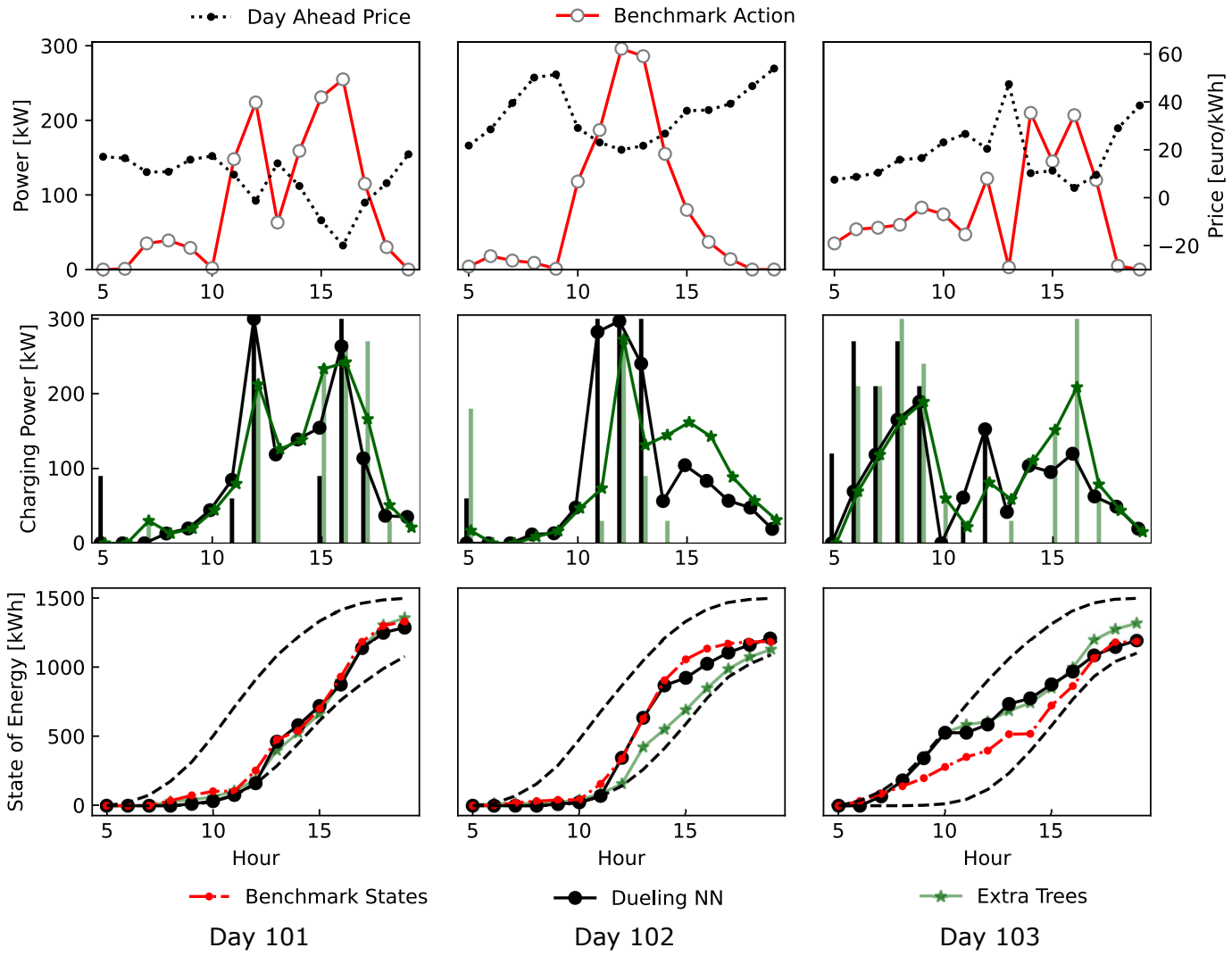

Figure 5. Comparison between best performing agents for both regression techniques over the 3 validation days. The top three sub-figures indicate the price profiles used for the 3 validation days along with optimal actions computed using Eq. (A29). The middle three sub-figures show the actions taken by the agents (u_t), indicated by the bars along with the actual power drawn by the fleet (u_t^{phys}) indicated by the lines, as given by Eq. (13). The bottom three sub-figures present the aggregate states followed by each of the agent during the operational period along with the benchmark states computed using Eq. (A29).

state and action comparisons between the best performing agents for both regression techniques.

For day 101, it can be observed that both the EXTRA trees and dueling neural networks based agents are able to produce a control policy close to the optimum policy and consequently, the aggregate states are close to the benchmark states. However, for day 102, the EXTRA trees based agent is slightly inferior with its policy computations and for day 103, both EXTRA trees and dueling neural networks based agents are not able to correctly follow the optimum actions. This is reflected in the aggregate states graph, where these agents cannot follow the benchmark states. While for day 102, the inferior performance of the EXTRA trees based agent can be attributed to a single sub-optimal action taken for time slot 11, for day 103, both agents tend to take rapid charging decisions for the first few time slots (6-9) and miss out on the lower prices at the end of the day (time slots 14-16). This sub-optimality can be attributed to marginal errors in approximating the Q-function.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 March 2021 doi:10.20944/preprints202103.0592.v1

Table 4: Performance of Best Performing Agents for the 3 validation days

	Dueling Neural Networks	EXTRA Trees
Day 101	0.72	0.77
Day 102	0.70	0.66
Day 103	0.51	0.57

5.2. Objective 2: Maximizing Self Consumption

Here, the primary objective of the agent is to maximize self-consumption from a local solar PV installation whilst respecting the individual charging constraints of each EV. For this scenario, a 150KWp PV installation is assumed that follows the generation pattern of a PV installation representative for an area located in Belgium. Additionally, a price asymmetry is assumed, where the price of selling power to the grid is assumed to be 20% less than the day ahead price for that hour. This assumption incentivizes the agent to consume the available solar power completely whenever possible. In case of deficits (or excess), the agent minimizes the cost of energy during the day. The training period includes a total of 10 different price and PV power profiles over the training period obtained from [25]. The performance of both regression models was then tested on the 3 validation days defined in Section 4.3. The comparison between performances of all 20 agents of both regression techniques for the 3 validation days is presented in Fig. 6.

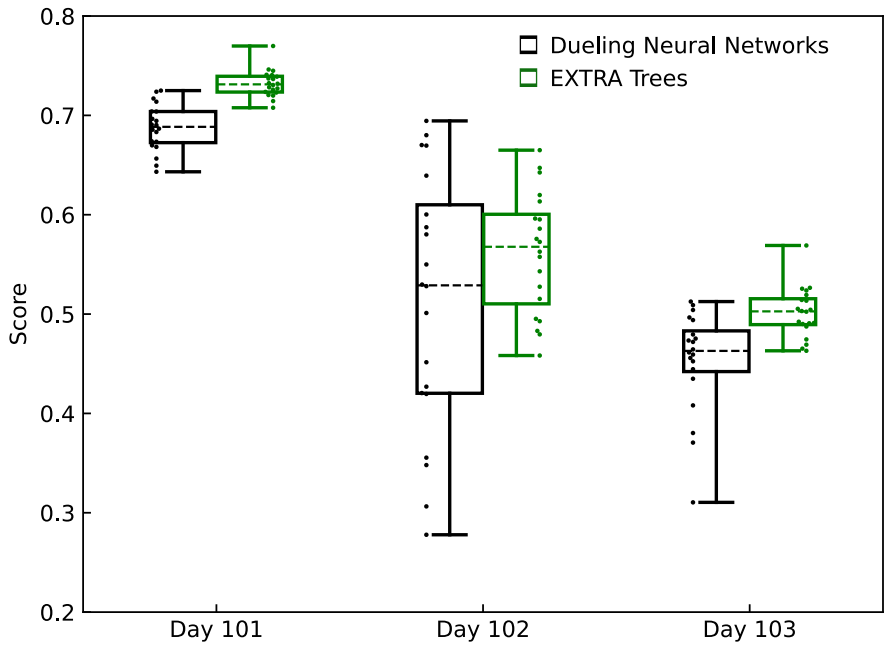

Figure 6. Box Plots for Maximising Self Consumption

In Fig. 6, we observe that for all 3 validation days, the performance of agents using EXTRA trees regression technique is slightly better than those using dueling neural networks. For days 101 and 103, EXTRA trees-based agents perform better than dueling neural networks as shown in Table 4. Further, similar to the case of cost minimization, the performance of EXTRA trees-based agents is more stable as compared to the dueling neural networks based agents. Table 4 shows the performance of best performing agents for both regression techniques. For this objective, best performing agents for both regression techniques achieve a cost reduction of more than 80% compared to the

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 March 2021 doi:10.20944/preprints202103.0592.v1

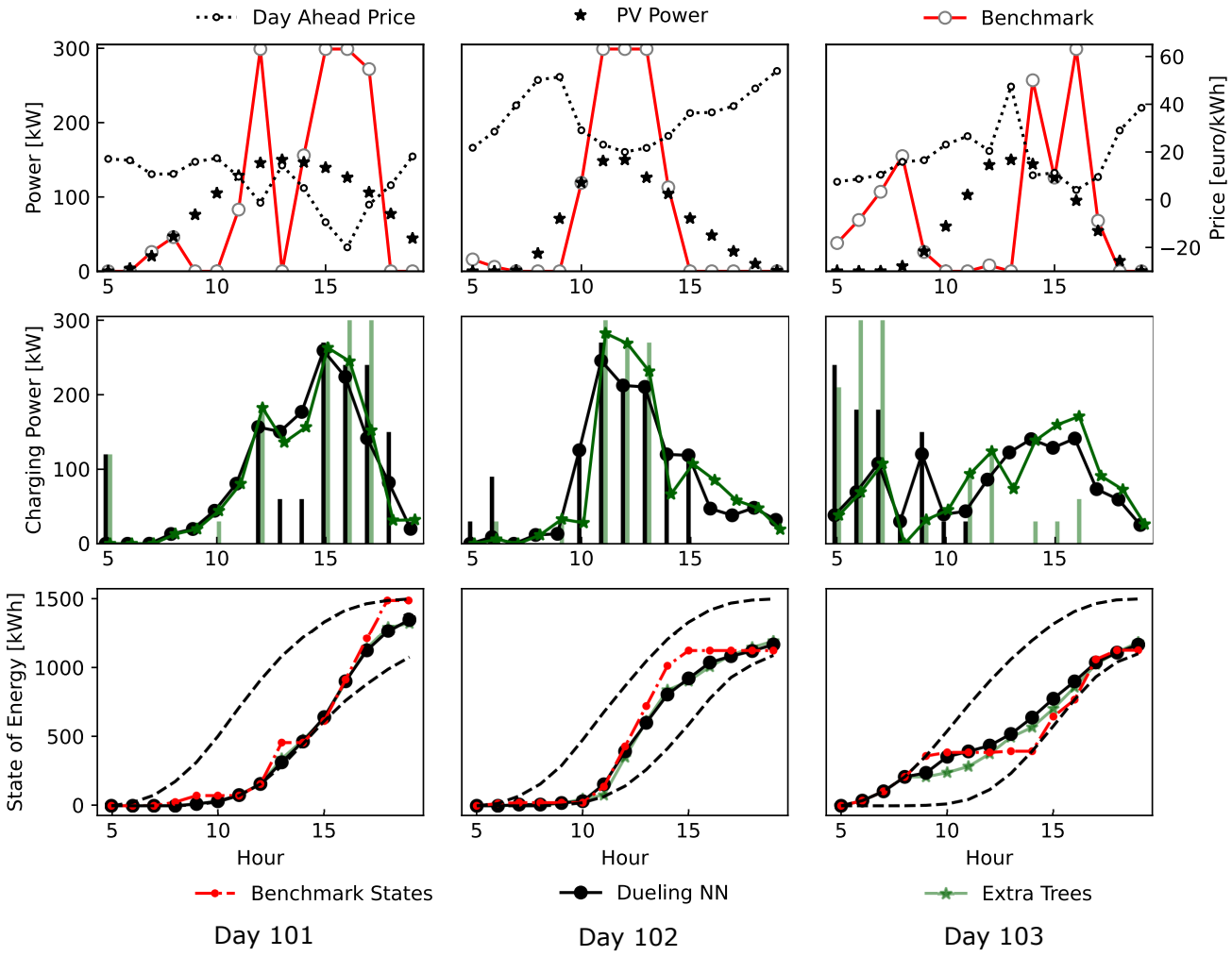

Figure 7. Comparison between best performing agents for both regression techniques over the 3 validation days. The top three sub-figures indicate the price and PV profiles used for the 3 validation days along with optimal actions computed using Eq. (31). The middle three sub-figures show the actions taken by the agents (u_t), indicated by bars along with the actual power drawn by the fleet (u_t^{phys}) indicated by lines. The bottom three sub-figures present the aggregate states followed by each of the agent during the operational period along with the benchmark states computed using Eq. (31).

business-as-usual case (shown in Table 2). Figure 7 shows the state and action comparisons between the best performing agents for both regression techniques.

For day 101 and 102, both agents have determined control actions that closely follow the optimum actions. Consequently, the aggregate state graph follows the benchmark states. Contrary to this, for day 103, actions taken by both agents deviate from the optimum actions. For day 103, both agents chose to charge with higher power at the beginning of the day (time slots 5-7) in the absence of any solar generation but for a lower price. Similarly, both agents chose to not charge at all for the later part of the day (time slots 11-17) even though PV generation is high and the price is low. Further, for these time slots, the agent actions are overridden by the environment and this can be observed by the deviation between the bar graphs and the lines. This signifies improper learning behavior and can be attributed to inferior regression performance by both techniques.

6. Conclusions

The results obtained in Section 5 indicate that an RL agent based on dueling neural networks can potentially reduce the charging cost incurred by an EV fleet operator by at least 60 to 80% as compared to the case of charging EVs as soon as they arrive (Tables

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 March 2021 doi:10.20944/preprints202103.0592.v1

1, 2). This is a significant reduction in cost whilst satisfying charging requirements of each EV and without the use of explicit knowledge about EV parameters like arrival and departure times or user driving profiles.

Further, for the two objectives considered in this paper, the scores obtained by agents using EXTRA trees regression techniques have less variance (i.e., are more stable) and are consistently higher than most dueling neural networks-based agents. While comparing the best performing agents, dueling neural networks can outperform EXTRA trees for some scenarios, but the dueling neural networks-based agents exhibit high variance in scores and lack stability in the training process. Hence it can be concluded that while dueling neural networks provide fairly accurate results, the resiliency of EXTRA trees technique makes it a go-to regression technique for such problems and information settings.

6.1. Future Work

With this research work, we investigated the application of dueling neural networks in an RL agent for deriving optimum charging decisions in an EV fleet problem. A significant issue in this approach is the high variance with scores obtained by agents using dueling neural networks. This lack of stability can be attributed to sub-optimal learning and the need for more data during the training process. Hence, future work will involve leveraging results from these experiments and focus on improvements towards two main aspects: (i) improving the neural network architecture and, (ii) improving the quality of training data. For the first, we plan to investigate model-based approaches to enrich the neural network with explicit information about the environment, specifically focusing on embedding model information in the neural network architecture. For the later, the focus would be on obtaining stable training performance over a low amount of data. We aim to study the model-based exploration techniques, including the use of control techniques like MPC for guiding the exploration in RL agents. Such techniques can help avoid exploring redundant states, thus narrowing the stat-action space considerably. Besides exploration techniques, using adversarial sampling and ensemble based techniques would also be investigated for stable training of the agents.

Author Contributions:

All authors contributed to this research work. Conceptualization, Bert Claessens; Formal analysis, Gargya Gokhale; Funding acquisition, Chris Develder; Investigation, Gargya Gokhale; Software, Gargya Gokhale; Supervision, Bert Claessens and Chris Develder; Visualization, Gargya Gokhale; Writing – original draft, Gargya Gokhale; Writing – review & editing, Bert Claessens and Chris Develder.

Funding: Part of the research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme for the projects BRIGHT (https://www.brightproject.eu/) - grant agreement no. 957816) and RENergetic (http://www.renergetic.eu/ - grant agreement no. 957845).

Acknowledgments: A major part of this project was carried out at Eindhoven University of Technology (TU/e) as a part of Master Thesis work of G.G, for the InnoEnergy Smart Electrical Networks and Systems program. The authors would like to thank the Electrical Energy Systems group at TU/e and the graduation project committee members including Prof. Mikael Amelin (KTH) and Prof Maarten Schoukens (TU/e).

Conflicts of Interest: The authors declare no conflict of interest.

Appendix G. Theoretical Benchmark

A theoretical benchmark is calculated to compare the performance of both regression models. This theoretical benchmark is formulated with the objective of minimizing the reward obtained over total time.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 March 2021 doi:10.20944/preprints202103.0592.v1

Appendix G.1. Minimising Cost

This theoretical benchmark is formulated as a mixed integer linear program (MILP) with the objective of minimizing the cost of energy during the charging process for each EV and then aggregating it for the entire fleet. The optimisation problem for an EV i is formulated as shown in Eq. (A29).

$$\begin{aligned} C_i = \underset{u_{i,t}}{\text{minimize}} \quad & \sum_{t=1}^{T-1} \lambda_t (x_{i,t+1}^{\text{phys}} - x_{i,t}^{\text{phys}}) \\ \text{subject to} \quad & \\ & x_{i,t+1}^{\text{phys}} = x_{i,t}^{\text{phys}} + u_{i,t}\Delta t \qquad u_{i,t} \in U \\ & E_{i,t}^{\min} \leq x_{i,t}^{\text{phys}} \leq E_{i,t}^{\max} \\ & \forall t \in \{1, \ldots, T\} \end{aligned} \tag{A29}$$

C_i represents the minimum cost of charging for EV i. The decision variables in the optimisation problem in Eq. (A29) are discrete integer actions, 0 and 3. The benchmark is obtained by aggregating over optimal values (C_i) for all EVs in the fleet as shown in Eq. (A30).

$$\text{Benchmark Value} = \sum_{i=1}^{N_{\text{ev}}} C_i \tag{A30}$$

Table 1 presents the benchmark values, total rewards for the business-as-usual case and the rewards obtained using the best performing agents for the 3 validation days described in Section 4.3. These values are used to calculate the scores of the trained agents and compare their performances.

Table 1: Minimizing Cost: Performance of Best Performing Agents for the 3 validation days

	Benchmark Value €	Total Reward w/o Controller €	Total Reward Models €	
			Dueling Neural Networks	EXTRA Trees
Day 101	-3194	13015	-2059	-2221
Day 102	30405	49906	34110	35670
Day 103	15157	29952	19743	20335

G.2. Maximising Self Consumption

This benchmark is formulated as a non-linear optimisation problem as shown in Eq. (A29).

$$\begin{aligned} \underset{u_{i,t}}{\text{minimize}} \quad & \sum_{t=1}^{T-1} \rho_t (x_{i,t}^{\text{phys}}, u_t^{\text{phys}}) \\ \text{subject to} \quad & \\ & x_{i,t}^{\text{phys}} = x_{i,t-1}^{\text{phys}} + u_{i,t}^{\text{phys}}\Delta t \qquad u_{i,t}^{\text{phys}} \in U \\ & E_{i,t}^{\min} \leq x_{i,t}^{\text{phys}} \leq E_{i,t}^{\max} \quad \forall t \in \{1, \ldots, T\} \\ & u_t^{\text{phys}} = \sum_{i=1}^{N_{\text{ev}}} u_{i,t}^{\text{phys}} \\ & x_{i,0}^{\text{phys}} = 0 \end{aligned} \tag{31}$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 March 2021 doi:10.20944/preprints202103.0592.v1

Table 2: Maximizing Self-Consumption: Performance of Best Performing Agents for the 3 validation days

	Benchmark Value €	Total Reward w/o Controller €	Total Reward Models €	
			Dueling Neural Networks	EXTRA Trees
Day 101	-10970	7894	-5688	-6631
Day 102	2418	23724	8809	9662
Day 103	-5006	11343	3005	2024

Here, $\rho_t(x_{i,t}^{\text{phys}}, u_t^{\text{phys}})$ is given by

$$\rho(u_{1,t}^{\text{phys}}, ., u_{N_t^{\text{con}},t}^{\text{phys}}, \lambda_t, \lambda_t^{\text{PV}}, P_t^{\text{PV}}) = \begin{cases} \lambda_t P_t^{\text{net}} & : 0 \leq P_t^{\text{net}} \\ \lambda_t^{\text{PV}} P_t^{\text{net}} & : P_t^{\text{net}} < 0 \end{cases}$$
$$P_t^{\text{net}} = \sum_{i=1}^{N_t^{\text{con}}} u_{i,t}^{\text{phys}} - P_t^{\text{PV}}, \tag{32}$$

where, N_t^{con} represents the number of EVs connected at time t. Eq. (A29) is solved using the differential evolution optimisation algorithm [26].

The benchmark values, rewards for business-as-usual case and best performing agents are shown in Table 2.

Table 3: Model parameters

Parameter	Value
Number of Outputs	11
Activation Function	Hyperbolic Tangent
Loss Function	Mean Squared Error
Optimiser	Adam
Dropout	0.1
Number of Neurons per layer	128
Hidden Layers	6 layers

H. Performance Comparison with Deep Neural Network

This section presents the results of simulations (described in Section 4) for vanilla deep neural network. A vanilla deep neural network implies a fully connected neural network with more than one hidden layers. The neural network parameters after hyperparameter tuning are shown in Table 3.

H.1. Objective 1: Minimizing Cost

The agents trained using dueling neural networks, EXTRA Trees and Vanilla deep neural networks are compared based on the scores obtained for the test cases described in Section 5. Figure 8 shows the results of this comparison. We observe that the agents trained using vanilla deep neural network perform much worse than the other two classes of agents. Additionally, similar to dueling neural networks, the performances of these agents are less stable.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 March 2021 doi:10.20944/preprints202103.0592.v1

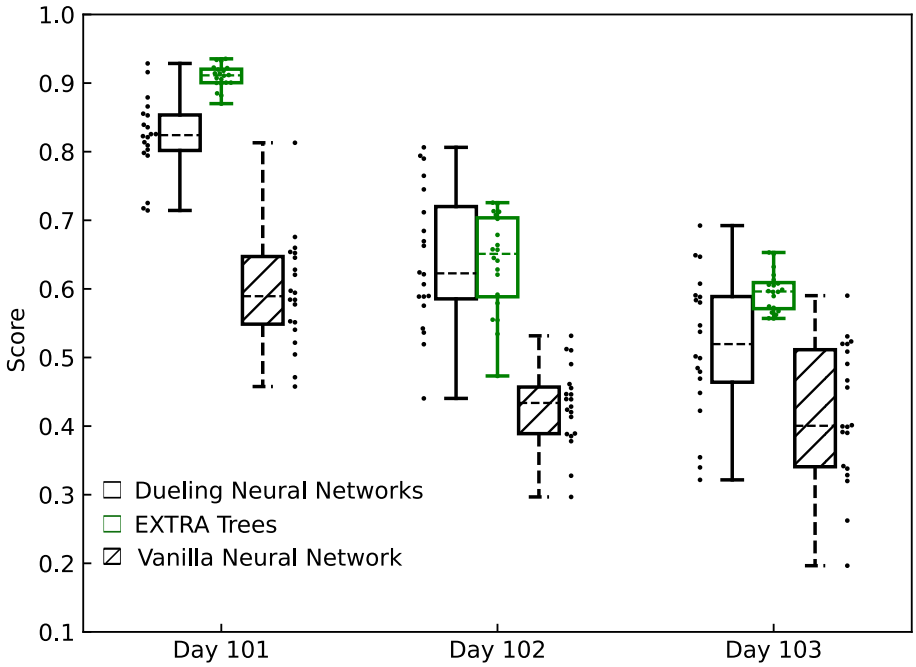

Figure 8. Comparison of performance of all agents for the validation scenarios and 3 regression techniques. The box plot represents the boundary of first and third quartiles, while the dotted line represents the median of the scores corresponding to the 20 agents. The dots depict the performances of individual agents.

H.2. Objective 2: Maximizing Self Consumption

Here, The agents trained using dueling neural networks, EXTRA Trees and Vanilla deep neural networks are compared based on the scores obtained for the test cases described in Section 5 and for the objective of maximizing self consumption. Figure 9 shows the results of this comparison. Similar to the previous case of Cost Minimization, it can be observed that the agents trained using vanilla deep neural network perform much worse than the other two classes of agents.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 March 2021 doi:10.20944/preprints202103.0592.v1

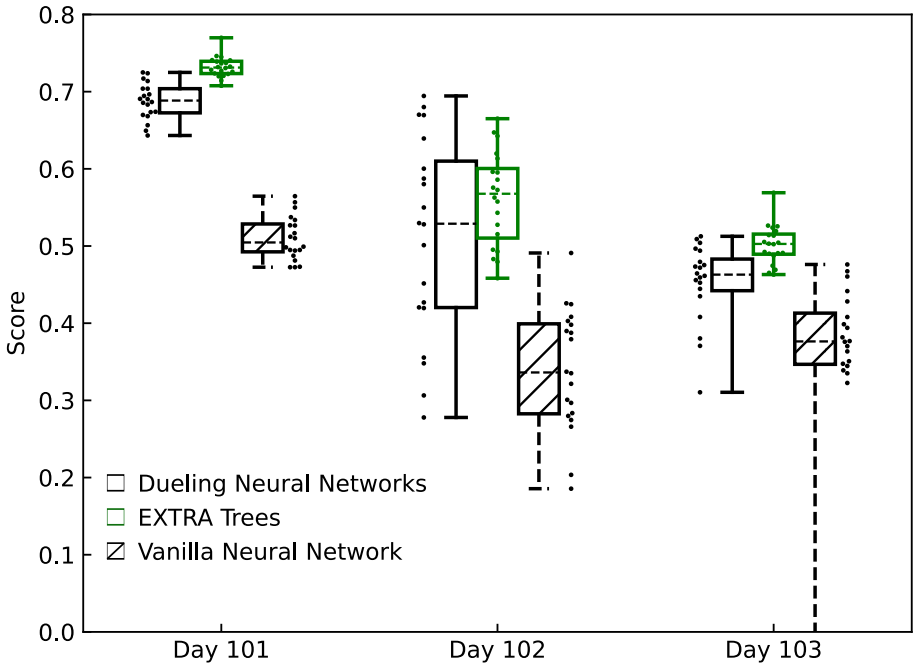

Figure 9. Comparison of performance of all agents for the validation scenarios and 3 regression techniques. The box plot represents the boundary of first and third quartiles, while the dotted line represents the median of the scores corresponding to the 20 agents. The dots depict the performances of individual agents.

Both Fig. 8 and Fig. 9 indicate that the vanilla neural network based agents are inferior in their performance. This is primarily due to inaccuracies in approximating the Q-function for the two objectives. Further, these results also indicate that comparatively, dueling neural networks based agents are better at approximating this Q-function due to the explicit streams for value and advantage function estimations.

References

1. Global EV Outlook 2020, IEA, Paris. https://www.iea.org/reports/global-ev-outlook-2020, accessed on 25 December 2020.
2. Hildermeier, J.; Kolokathis, C.; Rosenow, J.; Hogan, M.; Wiese, C.; Jahn, A. Smart EV charging: A global review of promising practices. *World Electric Vehicle Journal* **2019**, *10*, 80.
3. Hu, J.; Morais, H.; Sousa, T.; Lind, M. Electric vehicle fleet management in smart grids: A review of services, optimization and control aspects. *Renewable and Sustainable Energy Reviews* **2016**, *56*, 1207–1226. doi:10.1016/j.rser.2015.12.014.
4. Di Giorgio, A.; Liberati, F.; Canale, S. Electric vehicles charging control in a smart grid: A model predictive control approach. *Control Engineering Practice* **2014**, *22*, 147–162. doi:10.1016/j.conengprac.2013.10.005.
5. Ernst, D.; Glavic, M.; Capitanescu, F.; Wehenkel, L. Reinforcement learning versus model predictive control: A comparison on a power system problem. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* **2009**, *39*, 517–529. doi:10.1109/TSMCB.2008.2007630.
6. Sutton, R.S.; Barto, A.G. *Reinforcement Learning*, second ed.; The MIT Press, 2018.
7. Ernst, D.; Glavic, M.; Geurts, P.; Wehenkel, L. Approximate value iteration in the reinforcement learning context. application to electrical power system control. *International Journal of Emerging Electric Power Systems* **2005**, *3*, 1–35. doi:10.2202/1553-779X.1066.
8. Buşoniu, L.; Ernst, D.; De Schutter, B.; Babuška, R. Approximate reinforcement learning: An overview. 2011 IEEE symposium on adaptive dynamic programming and reinforcement learning (ADPRL). IEEE, 2011, pp. 1–8.
9. Wang, Z.; Schaul, T.; Hessel, M.; Van Hasselt, H.; Lanctot, M.; De Frcitas, N. Dueling Network Architectures for Deep Reinforcement Learning **2016**. *4*, 2939–2947, [1511.06581].
10. Arif, A.I.; Babar, M.; Ahamed, T.P.; Al-Ammar, E.A.; Nguyen, P.H.; Kamphuis, I.G.; Malik, N.H. Online scheduling of plug-in vehicles in dynamic pricing schemes. *Sustainable Energy, Grids and Networks* **2016**, *7*, 25–36. doi:10.1016/j.segan.2016.05.001.
11. Dimitrov, S.; Lguensat, R. Reinforcement Learning Based Algorithm for the Maximization of EV Charging Station Revenue. *Proceedings - 2014 International Conference on Mathematics and Computers in Sciences and in Industry, MCSI 2014* **2014**, pp. 235–239, [1407.1291]. doi:10.1109/MCSI.2014.54.
12. Li, H.; Wan, Z.; He, H. Constrained EV Charging Scheduling Based on Safe Deep Reinforcement Learning. *IEEE Transactions on Smart Grid* **2020**, *11*, 2427–2439. doi:10.1109/TSG.2019.2955437.

13. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; Petersen, S.; Beattie, C.; Sadik, A.; Antonoglou, I.; King, H.; Kumaran, D.; Wierstra, D.; Legg, S.; Hassabis, D. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. doi:10.1038/nature14236.
14. Wan, Z.; He, H.L.H.; Prokhorov, D. Model-Free Real-Time EV Charging Scheduling Based on Deep Reinforcement Learning. *IEEE Transactions on Smart Grid* **2018**, *10*, 5246–5257. doi:10.1109/TSG.2018.2879572.
15. 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*, 1950.
16. Vandael, S.; Claessens, B.; Ernst, D.; Holvoet, T.; Deconinck, G. Reinforcement Learning of Heuristic EV Fleet Charging in a Day-Ahead Electricity Market. *IEEE Transactions on Smart Grid* **2015**, *6*, 1795–1805. doi:10.1109/TSG.2015.2393059.
17. Geurts, P.; Ernst, D.; Wehenkel, L. Extremely randomized trees. *Machine Learning* **2006**, *63*, 3–42. doi:10.1007/s10994-006-6226-1.
18. Vandael, S.; Claessens, B.; Hommelberg, M.; Holvoet, T.; Deconinck, G. A scalable three-step approach for demand side management of plug-in hybrid vehicles. *IEEE Transactions on Smart Grid* **2013**, *4*, 720–728. doi:10.1109/TSG.2012.2213847.
19. Sadeghianpourhamami, N.; Deleu, J.; Develder, C. Definition and evaluation of model-free coordination of electrical vehicle charging with reinforcement learning. *IEEE Transactions on Smart Grid* **2020**, *11*, 203–214, [1809.10679]. doi:10.1109/TSG.2019.2920320.
20. Mbuwir, B.V.; Spiessens, F.; Deconinck, G. Benchmarking regression methods for function approximation in reinforcement learning: heat pump control. 2019 IEEE PES Innovative Smart Grid Technologies Europe (ISGT-Europe), 2019, pp. 1–5. doi:10.1109/ISGTEurope.2019.8905533.
21. Busoniu, L.; Babuska, R.; De Schutter, B.; Ernst, D. *Reinforcement learning and dynamic programming using function approximators*; Vol. 39, CRC press, 2010.
22. Ernst, D.; Geurts, P.; Wehenkel, L. Tree-Based Batch Mode Reinforcement Learning. *Journal of Machine Learning Research* **2005**, *6*, 503–556.
23. Nissan Leaf, EV Database. https://ev-database.org/car/1106/Nissan-Leaf, accessed on 03 July 2020.
24. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; Duchesnay, E. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* **2011**, *12*, 2825–2830.
25. ENTSO-E Transparency Platform. https://www.transparency.entsoe.eu/dashboard/show, accessed on 25 December 2020.
26. Storn, R.; Price, K. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization* **1997**, *11*, 341–359.