Machine learning approach towards remote diagnostics and repair of electric vehicle charging processes

(How) can electric vehicles learn to deal with faulty charging behaviour?

Dipl.-Ing. K. Renatus, Dipl.-Ing. A. Unger, Prof. Dr.-Ing. B. Bäker,
Dresden Institute of Automobile Engineering,
Technical University of Dresden;
Dr.-Ing. O. Manicke, Dr. Ing. h.c. F. Porsche AG, Weissach

Abstract

Electric vehicles have become a focus technology, promising a sustainable future mobility. Besides sufficient availability of charging infrastructure, ensuring functioning charging plays a crucial role for the success of e-mobility. This faces several challenges as related processes are more complex compared to refilling fuel of a conventional vehicle. Worldwide, different charging technologies, infrastructures and standards are being developed continuously. Realistically, not all can be considered during production. Besides, deviations from standards alongside implementation failures can cause incompatibility of vehicle and infrastructure. However, over-the-air updates enable the adaption of connected cars during their usage. Therefore, this paper firstly suggests using machine learning to allow vehicles to deal with certain unknown charging behaviour through automated parameterisation. Various system parameters on both sides can cause states which lead to an incomplete charging sequence. An intelligent agent shall learn parameters aiming for a complete charging process. Secondly, this paper proposes using data-driven digital twins of vehicles and infrastructure to perform backend simulations. The approach is demonstrated based on an exemplary use case and is evaluated via first simulation results. Future work will focus on expanding and improving the method, as well as implementing a real vehicle prototype.

1 Introduction

The global call for climate protection in combination with declining fossil resources and an increasing need for mobility, demand new concepts. Regarding the transportation sector, e-mobility supplied by green energy currently represents the most promising solution. Governments start to ban new vehicles with combustion engine in the future and automobile manufacturers steadily electrify their portfolios. According to estimates, 8 million electric vehicles (EVs) could be registered in Germany by 2030 [1]. In order to serve the rising number
of EVs, it is not only important to expand charging infrastructure, but also to ensure the reliability of charging processes. Compared to refilling fuel of a conventional vehicle, electric charging is much more complex. In this era of software, different types of plugs, for example, can be seen as a minor problem. A worldwide harmonisation of adequate standards, Charging Interface Initiative e.V. (CharIN) suggests the Combined Charging System (CCS) for example, seems the logical solution. But reality including market competition causes many challenges. Even though technologies evolve rapidly, electrification is still in its early stages. The slowly growing generation of EV owners often has to plan journeys carefully, because charging might not be possible. New charging technologies, infrastructures and standards are still being developed, which means not all can be considered during production. Besides, technical limitations alongside implementation failures can cause incompatibility of vehicle and infrastructure. A more detailed overview of possible failure causes is given in section 2. In order to prevent or fix charging errors, manufacturers and suppliers have to put in a lot of engineering effort throughout the product life cycle. As seen with other problems, Machine learning (ML) algorithms can help automating related processes. Section 3 discusses ways of how EVs could generally learn to avoid certain charging errors. Since reinforcement learning (RL) in particular has some solution-oriented advantages, section 4 focuses on introducing RL and transferring it to charging problems. As ML algorithms often require more computing power than standard control units or even high performance computers can provide, it makes sense to relocate complex computing processes to the backend. After detecting and diagnosing a faulty charging process, appropriate over-the-air software updates can be created, tested, validated and deployed remotely. Section 5 explains how data-driven digital twins can be used for backend simulations supported by intelligent algorithms.

2 Challenges regarding electric vehicle charging processes

2.1 Complexity due to diversity

To make electrification a success story from the user’s point of view, charging must be safe as well as comfortable and therefore simple, fast and most importantly error-free. A broken down vehicle in a remote area due to lack of charging thus can be seen as worst case scenario also for EV manufacturers. As electric charging is not harmonised worldwide, its reliability faces several challenges. The drafted standard IEC 62196 [2] refers to the four general conductive charging modes which mainly differ in terms of AC or DC charging, used cables and connectors, required protection and communication protocols. Regarding the faster DC charging for example, China introduced the GB/T standards, the CHAdeMO standard was developed in Japan, Tesla introduced their proprietary Superchargers in the US and CCS
began to spread from Germany across Europe and the US. A more detailed overview of charging infrastructures and standards is given by Parchomiuk et al. [3]. Consequently, some electric vehicle supply equipment (EVSE) and EV manufacturers started to implement multiple standards to satisfy location-dependent distribution. Another compromise solution with limited reliability are various adaptors for connectors [4]. Besides the existing variety, charging standards still evolve as recently seen with the ChaoJi standard, a harmonisation of GB/T and CHAdeMO. But just a matching connector does not guarantee successful charging as it is a distributed end-to-end process. The interaction of EV and EVSE can also increasingly include functional parts in the backend on both sides, e.g. during payment process. Hardware (HW) and software (SW) implementations, especially the communication controllers of EV and EVSE (EVCC, SECC), must conform to the used charging standard(s). But standards are mainly recommendations, thus real implementations can differ. On the one hand, the numbers of EVSE manufacturers, models and operators on the market still ascend. On the other hand, modern EVs and hybrid electric vehicles (HEVs) have increasingly complex electrical/electronic (E/E) architectures and networked functions, depending on original equipment manufacturers (OEM). The high-voltage (HV) system concept based on 400 or 800 V is only one example. Even within the portfolio of an OEM there are different model series, E/E-architectures, equipment variants as well as numerous HW and SW versions respectively continuous SW updates. Often more than one electronic control unit (ECU) is responsible for the charging functionality, which implies extensive dependencies that have to be validated. As mentioned, also charging itself is becoming more complex, especially regarding communication protocols. Whereas CHAdeMO uses a CAN-Bus, for example, CCS uses low level communication (pulse width modulated voltage signal on control pilot pin, mainly for AC charging) and high level communication (superimposed digital signal on control pilot and protective earth pin, for DC charging) as well as n-WLAN according to ISO 15118 [3] [5]. Innovative functionalities of smart charging, e.g. plug and charge (PnC) or bidirectional (keyword: smart grid) and wireless power transfer, already required protocol extensions as seen with the not yet released ISO 15118-20 extending ISO 15118-2 [5]. Summarised, the above mentioned complexity can disrupt compatibility and interoperability offering a lot of potential for errors during charging.

2.2 Analysis and focus of the problem

In order to eliminate or better avoid errors, it is necessary to analyse possible causes. Table 1 provides an overview of possible locations of error causes between EV and EVSE. For reasons of space, the EV side of the overview is based on a standard E/E architecture inspired by AUTOSAR [6] without considering modern high computing platforms with virtual ECUs.
combined with smart sensors and actuators. Due to different charging modes, charging cables can be assigned to EV or EVSE, whereby separate mode 4 cables with an in-cable-control-and-protection-device (ICCPD) can also cause SW-based errors.

Table 1: Brief overview of possible locations of error causes

<table>
<thead>
<tr>
<th>EV</th>
<th>Charging cable</th>
<th>EVSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>HW, other</td>
<td>other ECU</td>
<td>mode 1: separate</td>
</tr>
<tr>
<td></td>
<td>EVCC</td>
<td>mode 2: separate</td>
</tr>
<tr>
<td></td>
<td>connectivity ECU</td>
<td>with ICCPD</td>
</tr>
<tr>
<td></td>
<td>other ECU</td>
<td>mode 3: EVSE or</td>
</tr>
<tr>
<td></td>
<td></td>
<td>separate</td>
</tr>
<tr>
<td></td>
<td></td>
<td>charging station</td>
</tr>
<tr>
<td>BSWs, RTEs, SWCs</td>
<td>BSW, RTEs, SWCs</td>
<td>BSW ICCPD</td>
</tr>
<tr>
<td>other systems</td>
<td>BSW, RTEs, SWCs</td>
<td>RTE ICCPD</td>
</tr>
<tr>
<td></td>
<td>SWCs &amp; API</td>
<td>OS</td>
</tr>
<tr>
<td></td>
<td>SWCs &amp; API</td>
<td>SWCs &amp; API</td>
</tr>
</tbody>
</table>

As it may be possible to remotely diagnose, but not repair HW caused failures, e.g. defective connectors or insufficient memory of ECUs [7], the further focus is on SW. It is also assumed that basic software (BSW) and runtime environment (RTE) of involved controllers function properly, including the share of internal vehicle communication. On the application-related software component (SWC) level within an EV, there is the EVCC interacting with other ECUs. Connection ECUs can carry out backend (BE) connections via application programming interfaces (APIs) within the cellular network. Analogically, the EVSE can connect to a charging station management system (CSMS) in the backend, e.g. for value added services.

Regarding charging communication controlled by EVCC and SECC, Brosi [7] gives an extensive overview of fault categories, while introducing advanced conformity tests based on ISO 15118. SW errors in relation to component-based SW are discussed in detail by Mariani [8], whose taxonomy mainly differentiates syntactic, semantic, non-functional, connectors, infrastructure, topology and other faults. In the following, this paper will focus on errors that can be observed and possibly eliminated by the corrective action of parameterisation. Mostly, such errors can be assigned to semantical faults, which “[…] do not necessarily manifest with the impossibility of interacting, but may result in wrong results that manifest much later with a failure of another component”, according to Mariani [8]. Those can subsequently lead to non-functional (e.g. timeouts) or other faults. The need for parameterisation can underlie e.g. varying conditions of system and environment, human error or rarely ageing effects. In the automotive area, the process of adapting implemented vehicle functions by changing ECU-internal parameters via diagnostic tools is also called application. The following section will discuss how ML could be used to automate the application of charging.
3 Discussion of the term learning in relation to charging processes
After determining real and potential problems of charging, the question of possible solutions arises. In this emergent era of artificially intelligent (AI) solutions, an alternative question can be: (How) can electric vehicles learn to deal with, at least some, faulty charging behaviour?

3.1 Machine learning in short and preselection of approach
But firstly, it needs to be discussed why ML should be considered and what “learning” means in this matter. The AI pioneer Arthur Samuel stated that “Programming computers to learn from experience should eventually eliminate the need for much [...] detailed programming effort” [9]. Thus ML can be defined as the field of study that gives computers the ability to learn without being explicitly programmed. This implies the interpretation that related approaches allow machines to adapt to changing conditions. Regarding charging, errors due to varying conditions need to first be detected and then diagnosed in order to be repaired. Classical approaches, explicitly programmed, use data and rules applied to it as input aiming for an answer as output. ML approaches on the other hand, use data and thereof expected answers as input aiming to derive underlying rules as output [10]. From an EV point of view, the actual behaviour respectively rules of an EVSE are relatively unknown. Why not try to either let the EV learn the rules of an EVSE, or at least deal with them in order to self-adapt?

Superficially seen, there are three main types of ML: supervised learning, unsupervised learning and RL [11]. Supervised learning generally uses large sets of labelled data to train the relation between input and output variables, which can afterwards be used for classification or regression of unlabelled input. Unsupervised learning however uses unlabelled datasets as input and aims to solve clustering problems without active feedback about the trained similarities within clusters [10]. Regarding faulty charging, most likely there are not many datasets at all, there is no known solution and the aim is of a different nature.

RL shows the advantage of finding solutions to complex problems without large initial datasets. Essentially, RL is inspired by human learning based on trial and error. An agent sequentially interacts in discrete time steps \( t \) with its environment performing actions \( A_t \), which lead to new states \( S_{t+1} \) of the environment, a returned reward \( R_{t+1} \) indicates the success of each chosen action. The agent continually aims to find a strategy which results in positive experiences, here rewards [12]. This principle is shown in fig. 1 and will be recessed in section 4.
3.2 Clarification of the learning objective of reinforcement learning

Secondly, it needs to be discussed how a RL approach could be used for solving some of the aforementioned semantic errors during charging (cf. section 2). Looking at the product lifecycle, this goal is desirable during development phase, as well as during usage phase of an EV. The distributed system of EV and EVSE can be seen as the RL agent’s environment, in which the environment can be real or virtual. Enabling the interaction of agent and environment, in this case is not trivial as the following aspects have to be considered.

The implementation of the environment can be real, virtual or hybrid. Two major challenges arise, if real-time effects between EV and EVSE should be taken into account. Firstly, using real implementations massively restricts possible rates and runs of interactions with the agent. Imaginable standardised debug modes during charging do not (yet) exist. Secondly, functional safety must be assured in order not to endanger users. This can be solved, for example, by retaining parameter limits like found in ECU description files, determined during development. Alternatively, functional mock-up units (FMUs) of EV and/or EVSE can be implemented for testing. Virtual implementations for co simulations, can have different abstraction levels of which [13] gives an extensive overview regarding virtual ECUs (vECUs). Such can also be used within hardware-in-the-loop tests either representing real ECUs or rest-bus models. In order to sufficiently reproduce restrictedly realistic EV EVSE interaction behaviour, the use of software- or hardware-in-the-loop (SiL, HiL) test methods has to be according to the test use case and availability of suitable data, e.g. logging of communication. A hybrid implementation of the EV-EVSE environment is imaginable, e.g. a virtual EV interacting with real EVSE and vice versa. However, general interest today is to keep the demand of HW resources to a minimum by expanding the utilisation of virtual methods. In the context of this work it is initially assumed that virtual models of EV and EVSE are available as well as suitable data.

Different degrees of learning can be defined in terms of invested effort and time, as follows:

0) Rudimentary: EV avoids certain EVSE in future, after an incompatibility is detected.
1) One-off simple: EV tests predefined parameter limits (cf. above) once as possible solution. In the case of failure, rudimentary learning is applied.
2) One-off optimal: Agent performs parameterisation once aiming for an optimised solution, considering e.g. electric system protection for long remaining lifetime.

3) Mid-term: Agent learns for a defined period of time through interaction, optimising certain quantities until sufficient satisfaction.

4) Continuous: Agent continuously learns through interaction optimising certain quantities throughout the remaining lifetime of the EV.

The degree 0) is already used in EVs, degree 1) can be seen as backup which does not aim high. The degrees 3) and 4) do not apply to the problem in this work, an example is optimising the operating strategy of hybrid electric vehicles as looked at by Pertsch et al. [14]. In the scope of this work, a functioning parameterisation shall be found and updated, once or repeatedly if necessary. Therefore the following focuses on degree 2) of learning.

Looking at the fact that RL agents can be set up to follow different objectives, this paper suggests two generalised approaches towards diagnostics and repair of charging errors:

a) indirectly solution-oriented approach
This approach follows the principle that an agent shall deliberately cause errors through parameterisation. A similar idea is mentioned by Pertsch et al. [14]. One purpose could be to monitor the environments simulated behaviour in order to afterwards compare it with real faulty behaviour from field data. This way relevant monitored quantities could help diagnosing root causes of errors, if matching behaviour was reproduced. A second purpose could be to induce and thereby predict unseen errors, which can be used to expand future test coverage.

b) directly solution-oriented approach
This approach follows the principle that the agent shall directly find a positive outcome through parameterisation, meaning a functioning environmental behaviour. That way it would be possible to combine diagnosing with repairing a certain error, as the relevant parameter(s) is/are identified as well as quantified simultaneously. Because this implies opportunities regarding more effectiveness and less risk, the following will focus on the second approach.

4 Reinforcement learning approach for parameterisation
4.1 Brief introduction to reinforcement learning
Referring to fig. 1, the RL problem of choosing an action $a$ in a state $s$ can be formulated as finite Markov decision process. That corresponds to the assumption that the transition from state $S_t$ at time $t$ to subsequent state $S_{t+1}$ only depends on $S_t$ and $A_t$, neither on $S_{t-1}$ nor $A_{t-1}$. One challenge of RL algorithms is finding a balance between exploration and exploitation, both is necessary. Exploitation describes using the experience of which action leads to the highest reward in a certain state. This experience has to be gained by exploring new actions and
states, initially and in order to find possibly better actions. The agent’s decision strategy is represented by the policy $\pi$ which assigns possible actions to a known state. This policy is deterministic or stochastic, the second can be expressed as follows [12] [15]:

$$a \sim \pi(a|s)$$  \hspace{1cm} (1)

After each action the agent is returned a number, the reward $r$ which should be maximised in long-term. This can be expressed by the cumulative return $R$ within a finite period of time, in which the trajectory $\tau$ stands for the sequence of occurring states and actions [12] [15]:

$$R(\tau) = \sum_{t=0}^{T} R_t$$  \hspace{1cm} (2)

The value function $V$ describes which return the agent can expect in future by following the policy $\pi$, through assigning a value to every state (cf. (3)). An extension of the value function is the action-value function $Q$, which assigns a value to every state-action couple by following the policy $\pi$ when choosing an action $a$ [12] [15]:

$$V_{\pi}(s) = E_{\tau \sim \pi}[R(\tau) \mid s_0 = s]$$  \hspace{1cm} (3)

$$Q_{\pi}(s, a) = E_{\tau \sim \pi}[R(\tau) \mid s_0 = s, a_0 = a]$$  \hspace{1cm} (4)

In order to simplify the computation of value functions, the Bellman equation can be applied. It decomposes the value functions into the immediate reward and the discounted future values. The idea is to break the complex problem down into simpler sub-problems, following Bellman’s principle of optimality. It states that an optimal policy, whatever the initial state and initial action are, has the property that all subsequent actions must constitute an optimal policy as well. Accordingly, the aforementioned value functions can be transformed as follows [12] [15]:

$$V^*(s) = \max_a E_{\tau' \sim P}[r(s, a) + \gamma V^*(s')]$$  \hspace{1cm} (5)

$$Q^*(s, a) = E_{\tau' \sim P}[r(s, a) + \gamma \max_{a'} Q^*(s', a')]$$  \hspace{1cm} (6)

The equations above correspond to the aim of maximising the agent’s rewards, following an optimal (*) policy instead of just following a certain policy $\pi$. $s'$ and $a'$ describe the subsequent state and action. The discount factor $\gamma$ takes into account that the immediate reward can be more important than rewards earned in future. $P$ marks the probability of reaching state $s'$ when choosing action $a$ in state $s$ [12] [15].

Relatively simple problems can be solved using these equations, for example, by calculating discrete values transferred into tables which assign possible actions and rewards to certain states. More complex problems with thousands of states and more can reach the limits of related algorithms, as computing times increase unacceptably. Instead of using tables
assigning actions to states, approximated value functions in form of a weight vector $w$ can be used, as follows [12] [16]:

\[
\hat{v}(s, w) \approx V_{\pi}(s) \tag{7}
\]

\[
\hat{q}(s, a, w) \approx Q_{\pi}(s, a) \tag{8}
\]

The number of entries of the weight vector is much smaller than the state space, respectively state-action space. There are different ways of approximation, which can have state and/or action vectors as input. The weight vector could be, for example, a linear function in features of the state, a function computed by a decision tree, or more generally a function computed by a multi-layer artificial neural network [12]. As non-linear methods using neural networks have made major progress solving several problems in recent years, related algorithms are used for first simulations in this work. Roughly, RL can be divided into model-based and model-free algorithms, regarding the presence of an environmental model. As model-free approaches are easier to implement, such are used for first simulations. Advantage Actor Critic (A2C) follows a direct, proximal policy optimisation (PPO) an indirect policy optimisation. Deep Deterministic Policy Gradient (DDPG) combines policy optimisation with Q-learning. The extensive subject of RL is described more detailed in numerous sources like [12], [15] or [16].

4.2 Problem transfer and first results using model-free RL

For first simulations, the EV-EVSE environment is abstracted by introducing two adaptable parameters on the EV side. In combination these two parameters offer a discrete space of $10^4$ possible combinations. A black-box with pre-defined accepted parameter combinations represents a simplified EVSE. The RL agent receives positive rewards instead of negative rewards, once the EVSE signals a successful parameterisation (yes/no). Fig. 2 shows the average reward over episodes during an exemplary agent-environment interaction.

![Fig. 2: The agent-environment interaction of RL](https://doi.org/10.51202/9783181023846-163)
First simulations with the abstracted use case show that only DDPG manages to find a solution if both parameters have to be set in combination (cf. fig. 2). Simpler environments with only one parameter to be found can also be solved by A2C and PPO. Mostly, the DDPG algorithm needs less episodes to generate high rewards, but also needs more computing time.

5 Remote diagnostics & repair as a backend-service via digital twins

In order to provide sufficient computing power for algorithms like described above, connectivity combined with cloud technologies allow the relocation to the vehicle backend. Häberlein et al. suggest a central diagnostic instance for monitoring vehicle functions remotely [17]. This idea can be expanded by the concept of digital twins (DTs). Kritzinger et al. [18] see the DT as evolution from digital model via digital shadow based on the level of data integration. Accordingly, a DT offers automated data flows from its real twin and vice versa. Fuchs et al. [19] extend the understanding of an automotive DT, which involves of a virtual model offering services to a physical entity transferring data via connections, by a user interface. Services represent vehicle functions and need to be differentiated from backend-services. Fig. 3 shows a modified DT concept offering the backend-service of parameterisation to a connected EV.

![Fig. 3: Concept of data-driven digital twins for backend simulations](image-url)

Thinking in a big picture, a main advantage of this concept is that many connected vehicles worldwide can share field data as well as generated solutions. Such a solution can be a set of ECU-internal parameters enabling charging of certain EV types with mapped EVSE types. The concept offers a basis not only for remote diagnostics and repair of EV charging processes, but also for other backend-services concerning the perspective of “lifelong learning vehicles”.

https://doi.org/10.51202/9783181023846-163

Das Erstellen und Weitergeben von Kopien dieses PDFs ist nicht zulässig.
6 Conclusion and outlook

In this paper, current challenges regarding electric vehicle charging processes are identified. Possible errors between EV and EVSE are categorised in relation to the location of error causes. Focusing on semantic software faults which can be fixed by the corrective process of application, this work suggests using machine learning for automated adaption of ECU-internal parameters on EV side. As simpler methods, like brute force, are not very efficient, the investigation of reinforcement learning approaches is proposed. First simulations are based on an exemplary use case within an abstracted EV-EVSE environment for the RL agent to interact with. Three applied model-free algorithms based on artificial neural networks show, that the principle idea of using RL is applicable under certain conditions. Resuming from this positive conclusion, further resulting questions must be addressed. Firstly, simulation results must be expanded by improving used RL algorithms, by testing other RL algorithms, e.g. model-based, as well as comparing alternative algorithms. Secondly, related results must be transferred to charging in detail, tackling the challenge of providing trustworthy solutions for deterministic system behaviour. Thirdly, the concept of a digital twin based on virtual models, offering aforementioned algorithms as a service, needs to be proven by implementing a real vehicle prototype demonstrating remote diagnostics and repair.

References


