vehicles

MDPI

Article

Energy Management Strategy in 12-Volt Electrical System Based on Deep Reinforcement Learning

Ömer Tan [1,2,*], Daniel Jerouschek [1,2], Ralph Kennel [2] and Ahmet Taskiran [1]

[1] Department of System Integration and Energy Management, IAV GmbH, Weimarer Straße 10, 80807 Munich, Germany; daniel.jerouschek@iav.de (D.J.); ahmet.taskiran@iav.de (A.T.)
[2] Chair of High-Power Converter Systems, Technical University of Munich, Arcisstrasse 21, 80333 Munich, Germany; ralph.kennel@tum.de
* Correspondence: oemer.tan@iav.de

Abstract: The increasing electrification in motor vehicles in recent decades can be attributed to higher comfort and safety demands. Strong steering and braking maneuvers reduce the vehicle's electrical system voltage, which causes the vehicle electrical system voltage to drop below a critical voltage level. A sophisticated electrical energy management system (EEMS) is needed to coordinate the power flows within a 12-volt electrical system. To prevent the voltage supply from being insufficient for safety-critical consumers in such a case, the power consumption of several comfort consumers can be reduced or switched off completely. Rule-based (RB) energy management strategies are often used for this purpose, as they are easy to implement. However, this approach is subject to the limitation that it is vehicle-model-specific. For this reason, deep reinforcement learning (DRL) is used in the present work, which can intervene in a 12-volt electrical system, regardless of the type of vehicle, to ensure safety functions. A simulation-based study with a comprehensive model of a vehicle electric power system is conducted to show that the DRL-based strategy satisfies the main requirements of an actual vehicle. This method is tested in a simulation environment during driving scenarios that are critical for the system's voltage stability. Finally, this is compared with the rule-based energy management system using actual vehicle measurements. Concluding measurements reveal that this method is able to increase the voltage at the most critical position of the 12-volt electrical system by approximately 0.6 V.

Keywords: electrical energy management; energy management strategies; deep reinforcement learning; neural network; machine learning; 12-volt electrical system

Citation: Tan, Ö.; Jerouschek, D.; Kennel, R.; Taskiran, A. Energy Management Strategy in 12-Volt Electrical System Based on Deep Reinforcement Learning. *Vehicles* **2022**, *4*, 621–638. https://doi.org/10.3390/vehicles4020036

Academic Editor: Mohammed Chadli

Received: 1 May 2022
Accepted: 15 June 2022
Published: 20 June 2022

1. Introduction

In the course of the increased demands for comfort, safety and efficiency in motor vehicles over the last several decades, electronic units have replaced mechanically implemented functions [1]. In addition, completely new functions, such as electronic engine management systems to reduce fuel consumption and emissions, airbags and universal serial bus (USB) connections or infotainment systems, including smartphone interfaces, have become standard in automobiles [2]. Due to the rapid increase in electrification [3] in recent decades, modern motor vehicles are developing into the most technically complex consumer goods of all [4], not only because of new mobility concepts such as autonomous driving, but it can be assumed that this trend continues at a similar pace in the coming years [2].

However, safety-critical consumers such as electric power steering (EPS) assistance or electronic stability control (ESC) also require a high degree of power in extreme situations. This leads to a balancing process between guaranteeing the power supply to all consumers and ensuring that no critical voltage levels are undershot due to the brief activation of safety-relevant consumers [5].

In conventional vehicles, the alternator maintains the voltage in the 12-volt electrical system. If the alternator current is no longer sufficient—due, for example, to idling or because of a low engine speed—for maintaining sufficient supply to all consumers, this results in the vehicle electrical system voltage dropping to the battery voltage level. This means that the battery is supplying power to the consumers. As a result, the battery is responsible for attenuating voltage peaks in the vehicle electrical system voltage to protect sensitive electronic and electrical components [6]. The main risk is that the battery voltage is insufficient for maintaining safety-critical functions in extreme situations. The overlapping of comfort consumers and safety-critical high-power consumers can cause voltage dips that drop to 10V or less [7]. This problem is particularly important in autonomous vehicles [5].

An electrical energy management system (EEMS) coordinates the energy flows in a vehicle´s 12-volt electrical system. Here, the battery and the alternator energy act as energy sources [8]. A widespread strategy is to use predefined rules to briefly deactivate or switch down comfort consumers, such as seat heaters or the air-conditioning fan, in the event of low energy availability. This maintains safety-critical functions and prevents battery damage. However, this conventional or rule-based (RB) approach is time-consuming, and the preformulated rules are limited to specific vehicle types [9].

In order to demonstrate the disadvantages of rule-based systems, optimization-based and learning-based methods are researched [10]. In particular, the reinforcement learning (RL) method is used in many studies for the energy management system in hybrid electric vehicles (HEVs) [11–13], achieving improvements in fuel consumption [14]. However, these studies usually consider the HEV´s traction and not the individual components in the 12-volt electrical system. Due to the high degree of complexity of the electrical energy system, the conventional vehicle with an internal combustion engine (ICE) and a belt-driven alternator offer scope for electrical energy management.

In this paper, we present an energy management strategy in 12-volt electrical systems based on deep reinforcement learning. A test setup and multiple experimental implementations in different scenarios using different parameters show how DRL optimizes an EEMS in 12-volt electrical systems. The results of this paper show that DRL is suitable for taking over parts of the EEMS in 12-volt electrical systems in certain driving situations. For this purpose, we implemented three different agents trained with a detailed battery model and analyzed them in more detail. Subsequently, we compared the overall 12-volt electrical system with an actual driving scenario and real-life measurements. Here, the deep q-network agent (DQN) proved to be the most effective variant. When investigating voltage stability in the 12-volt electrical system, the DQN agent was able to keep the voltage reduction in critical situations lower than the rule-based variant by means of intelligent coordination of the electrical loads. Based on the conduction experiments, it was shown that RL-based methods optimized the coordination of current flows within a 12-volt electrical system, offering themselves as an alternative to previous rule-based systems.

2. Energy Management Strategy in 12-Volt Electrical System

The electrical energy management system in modern power networks comes as a measure to meet requirements. One of the reasons for the EEMS is the increasing number of consumers. Heat consumers such as heaters and air conditioning systems demand a high degree of continuous power [5]. Modern systems, such as brake-by-wire or electric power steering, are consumers that also require a high degree of power. Keeping these safety-relevant consumers properly supplied takes top priority. The EEMS ensures that energy is reliably provided for all components in the vehicle in all situations. This is performed as cost-effectively as possible, in an energy-efficient manner that saves installation space.

The most common EEMS measure mentioned in the literature is to switch off consumers [5,6,15–17]. When the load power exceeds the generator power, loads are degraded or switched off to force a positive balance again. However, these are not safety-critical loads. It is also important that passengers do not notice this degradation or shutdown. Thermal consumers are particularly suitable for this purpose, since heating elements are

generally very inert. As a result, vehicle passengers do not directly notice, for example, if the seat heating is briefly switched off.

If energy management is implemented using a ranking list, each consumer must be assigned a unique priority. Sorted according to this priority, the consumers are stored in a static ranking list. Prioritizing the consumers in the ranking list ensures that the most important consumers are given high priority. If the available power in the power network is insufficient for supplying all consumers, low-priority consumers are degraded or switched off.

In the rule-based approach, a system is implemented by using a set of predefined rules based on experience. Positive features are ease of use, robustness and good real-time performance [18]. However, the main disadvantage of using the system based on predefined rules is that adjusting and calibrating the rules with respect to the vehicle is time-consuming [9]. In addition, the rules cannot be transferred to other drive systems and vehicle models [19]. To overcome these disadvantages when applying machine learning techniques, learning-based approaches such as deep reinforcement learning are being explored.

3. Materials and Methods

3.1. Deep-Reinforcement-Learning-Based Energy Management Strategy

The deep-reinforcement-learning-based energy management strategy that was developed combines deep neural networks and conventional RL. The EEMS makes decisions based solely on the current system state. The rest of this section explains the theory of DRL, introduces the three agents developed and presents the environment.

3.1.1. Reinforcement Learning Background

An agent can be viewed as a software component that is capable of performing actions within an environment to achieve goals predefined by a user [20]. Similar to training a biological being, training an agent in the context of RL can be viewed as goal-directed learning based on trial and error [21], where the environment rewards or punishes the agent depending on the actions taken. A Markov decision process (MDP) can model such a decision problem. This forms the basis of an RL problem [22]. In particular, combining different deep learning techniques, such as using deep neural networks [23] and RL, has led to breakthroughs in the past decade [24–26]. The advantages of using RL are that in an environment that has a high-dimensional state space, error-prone manual feature selection becomes obsolete [27,28]. Specifically, RL agents have been developed in recent years that have found game-like solutions to master Atari games that beat highly skilled poker players [29–31]. In this context, the work of DeepMind research director David Silver´s team of developers stands out as groundbreaking. Their self-learning RL agents AlphaGo and AlphaStar were able to defeat the Go world champion Lee Sedol and to master the strategy game StarCraft II well [32].

3.1.2. Markov Decision Process

As mentioned, RL methods can be used to solve sequential decision problems. A decision problem can be mathematically modeled with an MDP. This process can be considered a finite tuple of the type (S, A, T, R, γ) [33], where:

- S is the state space;
- A is the actions space;
- $T: S \times A \times S \rightarrow [0, 1]$ is the transition function;
- $R: S \times A \times S \rightarrow R$ is the reward function;
- $fl \in [0, 1)$ is the discount factor.

Observations as well as agent actions are to be interpreted generically. However, agent actions can consist of controlling a robotic arm while receiving observations about an environment via sensors [21]. An agent interacts with its environment in that:

- The agent starts in a certain state $s_0 \in S$;
- It receives an initial observation $\omega_0 \in \Omega$;
- It executes an action at $a_t \in A$ each time $t \in T$ [33]. For each action:
- The agent receives a reward/punishment $r_{t+1} \in R$;
- The state changes from $s_t \in S$ to $s_{t+1} \in S$;
- The agent receives another observation $\omega_{t+1} \in \Omega$.

Figure 1 illustrates the interaction between agent and environment, and the state transitions and rewards resulting from actions. If the environment is fully known, it holds that the observation matches the state $\omega_t = s_t$ [33].

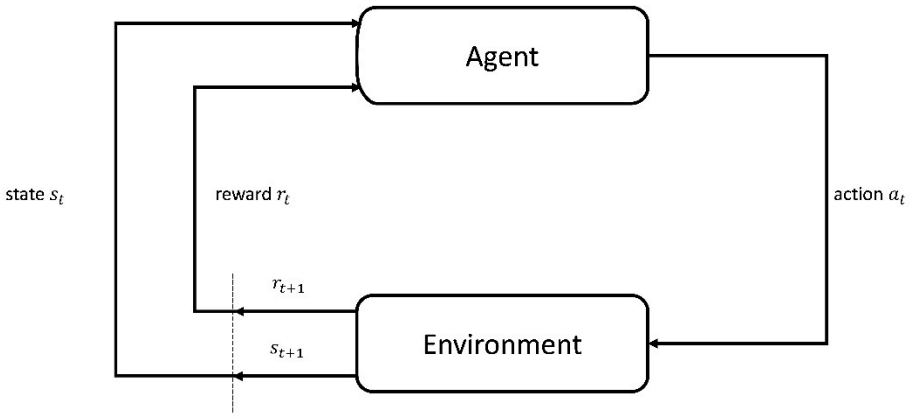


Figure 1. Interaction of an agent with an environment; a_t is the action taken by the agent at time t, s_t is the state of the environment at time t, s_{t+1} and r_{t+1} stand for the state and reward at time $t+1$.

In a finite MDP, the state and action space are finite sets, such that the random variables R and S are well-defined probability distributions depending on the prior state s_{t-1} as well as the prior action a_{t-1} [21]. This dependence satisfies the Markov property. As a result, the total history is irrelevant for upcoming actions, since previous interactions of the agent with the environment are clustered in the current state [21,33]. This property can be formalized for observations as follows:

$$\mathbb{P}(\omega_{t+1}|\omega_t, a_t) \doteq \mathbb{P}(\omega_{t+1}|\omega_t, a_t, \ldots, \omega_0, a_0) \quad (1)$$

The basis for the next observation is, thus, the current observation and action of the agent. The basis for the next reward also depends only on the current observation and action:

$$\mathbb{P}(r_{t+1}|\omega_t, a_t) \doteq \mathbb{P}(r_{t+1}|\omega_t, a_t, \ldots, \omega_0, a_0) \quad (2)$$

If state changes exist over time, the system is dynamic [34]. Since the MDP is abstract and flexible, it can be applied to many different problems and can include discrete time steps as well as random time intervals [21].

3.1.3. Policy

The agent´s actions are selected according to the policy π, which defines its behavior. Policies can be differentiated as follows:

- Deterministic: A deterministic policy is an assignment of a set of states to a set of actions: $\pi(\mathrm{s})$: $\mathrm{S} \rightarrow \mathrm{A}$. Here, actions a and $\pi(\mathrm{s})$, respectively, are always taken in state s [22].
- Stochastic: A policy is stated to be stochastic if it is described as follows: $\pi(\mathrm{s}, \mathrm{a})$: $\mathrm{S} \times \mathrm{A} \rightarrow [0, 1]$. Here, the probability that action a is taken in state s is denoted by $\pi(\mathrm{s}, \mathrm{a})$, respectively, as the probability distribution over actions A in state s [33].

Another way to distinguish policies from each other is to look at the affinity for risk when choosing actions [21].

- Greedy policy: The agent will always choose the first-best solution option, even if the solution might turn out to be particularly bad later on. Applying this type of policies is also called exploitation.
- Optimal policy: This policy always tries to find the best way. This is referred to as exploration. Even if the reward is increased in the short term, an alternative is preferred in order to find new possible solutions in the long term.
- ε -greedy policy: The- ε -greedy policy represents a compromise of the previously listed policies. In this case, a $\varepsilon \in [0, 1]$ is determined. Before executing an action, a random number is drawn from the same range of values. If the random number is greater than ε, an action is performed in terms of the greedy policy, otherwise, it is an exploratory action as in an optimal policy. Algorithm 1 is used as an illustration.

Algorithm 1: e-greedy policy.

```
1. Function GETPOLICY(EPSILON)
2.   randomNumber = getRandomNumber(0,1)
3.   If randomNumber > epsilon then
4.     Return greedyPolicy()
5.   Elseelse
6.     Return optimalPolicy()
7.     End ifend if
8. End function
```

3.1.4. Reward

Actions of an agent lead to rewards or punishments. Numerically, the difference does not matter; the goal is to maximize the expected reward [22]. The reward G at time step t is created by summing future rewards at each time step [21].

$$G_t \doteq r_{t+1} + r_{t+2} + r_{t+3} + \ldots + r_t = \sum_{k=0}^{T-1} r_{t+k+1} \tag{3}$$

For continuous tasks, the following applies:

$$G_t \doteq r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots + r_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \tag{4}$$

Since the individual reward is interdependent, it is also possible to rephrase:

$$\begin{aligned} G_t &\doteq r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 r_{t+4} + \ldots \\ &= r_{t+1} + \gamma\left(r_{t+2} + \gamma r_{t+3} + \gamma^2 r_{t+4} + \ldots\right) \\ &= r_{t+1} + \gamma G_{t+1} \end{aligned} \tag{5}$$

As $\gamma \in (0, 1)$ decreases over time with rewards, the importance of each reward decreases until it approaches a threshold value. In general, a $\gamma < 1$ is used to converge tasks to a final state. A $\gamma = 1$ corresponds to an environment which itself converges to a final state after many time steps, for example a chess game. For $\gamma = 0$, the agent is myopic, since, in this case, only the immediate reward is maximized [21]. Furthermore, the use of the discount factor corresponds to the geometric series, so that for $\gamma < 1$ [35]:

$$G_t = \sum_{k=0}^{\infty} \gamma^k = \frac{1}{1-\gamma} \tag{6}$$

3.1.5. Actor–Critic Methods

Policy gradient algorithms do not require a value function to select actions, but learn a policy directly through a vector parametrized with θ. In general, the policy is stochastic and denotes the probability of selecting action a in state s [21,35]:

$$\pi(a|s,\theta) = \Pr\{a_t = a|s_t = s,\ \theta_t = \theta\} \tag{7}$$

To determine the goodness of the policy, the performance measure $J(\theta)$ was used, which represents the discount expected rewards:

$$J(\theta) = \mathbb{E}z_\pi(s),\ \pi(a|s,\ \theta)\{G\} \tag{8}$$

$Z_\pi(s)$ denotes the state distribution of policy π. Via the gradient descent procedure, the gradient $J(\theta)$ was used to update the policy:

$$\theta_{t+1} = \theta_t + \alpha\nabla J(\theta)|\theta = \theta_t \tag{9}$$

Here, α denotes the learning rate. Algorithms that follow this approach are generally referred to as policy gradient methods and belong to the category of exclusively policy-based actor variants [36]. Since a pure policy method requires significantly fewer parameters than a value function, the use of an actor method can be advantageous [37]. These approaches can be extended to include value functions; consequently, they are referred to as actor–critic methods. Figure 2 illustrates the learning flow in these methods [38].

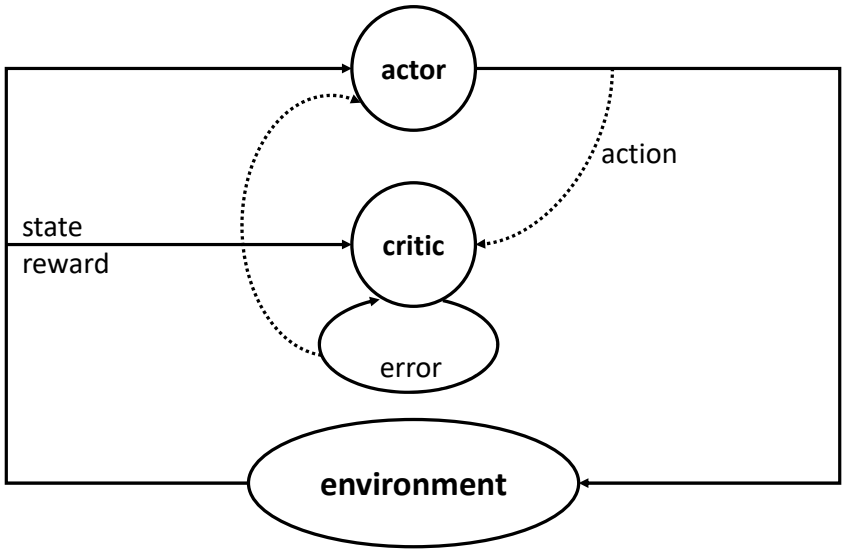


Figure 2. Actor–critic architecture.

In this case, the actor represents the policy and the critic the value function [39]. First, the actor interacts with its environment by performing an exploratory action, and then the critic responds by evaluating the action taken (policy evaluation). The value of the performed action in the respective state is compared to the expected long-term reward of the respective state using the critic´s value function. By adjusting the probability for the respective action, the policy is ultimately updated (policy improvement) [40].

3.1.6. Algorithmic Formulations of RL Approaches

In the following, the RL agents considered in this work were described in more detail. The algorithms presented as pseudocodes were mainly adapted from [21].

Deep Q-Learning

Basically, Q-learning (QL) falls under the category of temporal difference learning. The Q-learning algorithm is model-free and off-policy. The Q-table is updated with the

sum of the reward for the current state and with the maximum reward that can be achieved in a subsequent state. On the one hand, this is the reason for the off-policy property, but on the other hand, in large state spaces, this leads to the problem of exponentially increasing computation times with an increasing number of state variables.

To address the mentioned problem of large state spaces, function approximators are used as in deep Q-learning (DQL). In this case, no table was built, but the parameters of a neural network (NN) were optimized to maximize the expected long-term reward. Figure 3 illustrates DQL using an NN to approximate a function [41]. Here, the state was given as an input to the NN. The outputs of the system were the Q-values of all possible actions.

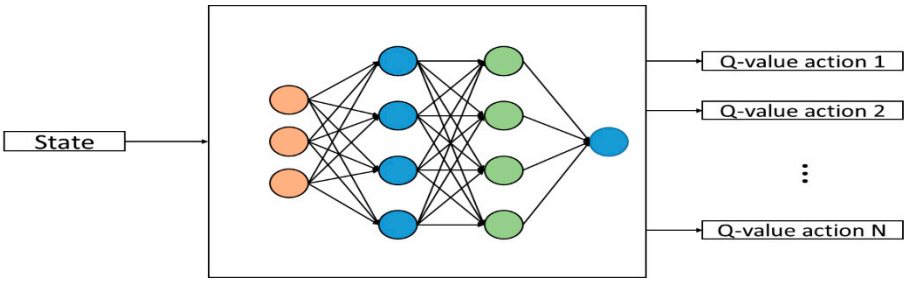


Figure 3. Deep Q-learning architecture with neural network in reference to [41].

In essence, the difference between conventional Q-learning and deep Q-learning is that Q-learning optimizes the values of a table and DQN optimizes the weights of a neural network. In order to further increase the learning stability or reduce oscillatory behavior in the allocation of rewards, another Q-network is mostly used [42]. This target network is identical in structure to the actual Q-network. The periodic copying of the weights of the original network to the target network stabilizes the training overall [22].

Policy Gradient

Another variant used in this work was presented below. This is the first and most fundamental variant of a policy gradient (PG) method: REINFORCE. In this algorithm, the degree of variance was determined using a stochastic policy π.

The environment was explored naturally by simply following the policy [36]. The product of the probability of an action $\pi(a_t|s_t, \theta)$ and the observed reward G after this action were performed to be maximized. As a result, the probability distribution was successively adjusted so that, optimally, after training was complete, the actions that promised the highest reward were selected [22].

Depending on the number of looking-ahead steps [43], fewer or more rewards were used to perform backpropagation. A distinction was determined here between one-step, the maximum number of steps in an episode (de facto Monte Carlo) and a compromise of the two (N-step) [22]. The Monte Carlo property of the REINFORCE algorithm in Algorithm 2 arose from using the reward at time t, which included all future rewards until the end of the episode [21].

Algorithm 2: REINFORCE Monte-Carlo-based policy gradient.

Input: a differentiable policy parameterization $\pi(a|s, \theta)$
Parameters: step sizes, $\alpha > 0$
Initialize policy parameter $\theta \in \mathbb{R}^{d'}$ (e.g., to 0)
For each episode **do**
    Generate an episode $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot \mid \cdot, \theta)$
    For each step t of episode $t = 0, 1, \ldots, T-1$ **do**
        $G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$
        $\theta \leftarrow \theta + \alpha \gamma^t G \nabla \ln \pi(A_t|S_t, \theta)$
    End
End

Actor–Critic

The one-step actor–critic (AC) algorithm shown in Algorithm 3 is considered an online learning procedure, since the extreme opposite of Monte Carlo was used [21]. The policy parameters θ were no longer computed using the reward G, but rather using the state value function (the critic) without including all future rewards of the particular episode.

Algorithm 3: Actor–critic.

Input: a differentiable policy parameterization $\pi(a|s,\theta)$
Input: a differentiable state value function parameterization $\acute{v}$(s,w)
Parameters: step sizes, $\alpha^{\theta} > 0, \alpha^{w} > 0$
Initialize policy parameter $\theta \in \mathbb{R}^{d'}$ and state value weights $w \in \mathbb{R}^{d}$ (e.g., to 0)
For each episode **do**
    Initialize S, the first state of the episode
    $I \leftarrow 1$
    While S is not terminal (**for** each time step):
        $A \sim \pi(\cdot|S,\theta)$
        Take action A, observe $S\prime, R$
        $\delta \leftarrow R + \gamma\acute{v}(S\prime, w) - \acute{v}(S, \omega)$ (if S is terminal, then $\acute{v}(s\prime,\omega) \doteq 0$)
        $w \leftarrow w + \alpha^{w}\delta\nabla\acute{v}(S, w)$
        $\theta \leftarrow \theta + \alpha^{\theta} I \delta \nabla \ln \pi(A|S, \theta)$
        $I \leftarrow \gamma I$
        $S \leftarrow S\prime$
    End
End

To reduce the risk of early, suboptimal convergence, a penalty term can be added to both policy gradient and actor–critic methods (entropy maximization) [44], which ensures that additional gradients are accumulated so that the objective function is minimized; exploratory actions are encouraged by a higher penalty term [30].

3.2. The Environment

In vehicle EEMS, the RL agent interacts with the environment, learns from past control experiences and continuously improves the control action. At each time step, the agent generates a set of control actions and observes an immediate reward. Thus, environment design plays a crucial role in the reinforcement learning process, since all the agent´s knowledge comes from its interaction with the environment. Therefore, it is important to abstract the problem without losing decision-critical parameters.

In this simulation-based study, a conventional vehicle electrical energy system was investigated. The model included a generator, a lead-acid battery and various loads such as seat heating or air conditioning. Figure 4 shows the electrical equivalent circuit of the simulated system. The parameters of the battery used are listed in Table 1. An alternator model with a maximum nominal current of 180 A was used. In addition, high-power consumers such as electric power steering assistance or dynamic stability control were implemented in the model as power curves.

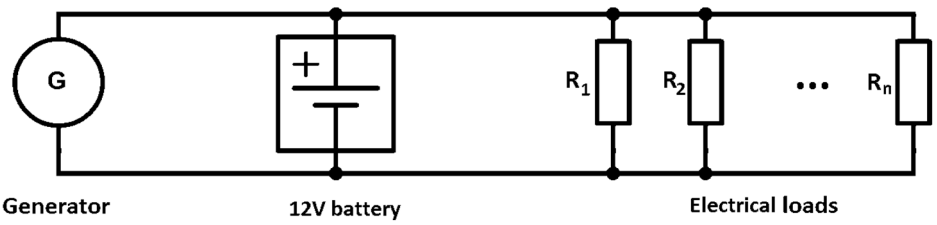


Figure 4. Electrical equivalent circuit of the electrical system.

Table 1. Specification of the tested battery.

Type	Lead-Acid AGM
Nominal Voltage	12 V
Nominal Capacity	70 Ah
Max. Discharge Current	720 A

4. Experimental Results

The experiments performed with the respective RL methods were explained below. Three agents (DQN, PG and AC) were used for the scenarios. The following points were considered:

- The takeover from the electrical consumers, activated by the passengers;
- Targeted consumer degradation or shutoff in critical states.

The aim was to show which agent was best suited to which parameters to intervene in an EEMS of a 12-volt electrical system in order to prevent battery damage and ensure safety functions. Specifically, comfort consumers were to be briefly downgraded or completely deactivated, but passenger-activated consumers should be maintained during stable system conditions.

4.1. Experiment Setup

The model used to determine the appropriate applicability of an RL agent for EEMS in a vehicle 12-volt electrical system was developed in MATLAB/Simulink 2020. The specification of the hardware used for the experiments performed is shown in Table 2.

Table 2. Specification of the hardware used for training.

Aspect	Metric
Operating System	Microsoft Windows 10 Enterprise
Processor	Intel® Xeon® W-2133 Central Processing Unit @3.60 GHz
Random Access Memory (RAM)	32 GB
System Type	64-Bit

4.2. Modeling Cost and Loss Function

The factor for the largest penalty of a comfort consumer was set as β. The penalties for the remaining consumers were created by looking at the largest difference in current consumption between two levels. If it was a consumer with more than two levels, the average differences were calculated. The penalties of the remaining consumers were then calculated as a function of the selected β.

This was to prevent the indirect prioritization of consumers by the agent. Simply stated, the agent preferentially degraded consumers which consumed a lot of current, but for which it would receive a penalty as large as for the other consumers. This would have the character of a rule-based system or a priority list, which should have been avoided in this study.

The penalties resulted from the following calculation: if μ_i was the (average) difference between the stages of a consumer i, μ_g was the largest (average) difference between two stages (across consumers). Factor k_i for the penalty of any i-th consumer was calculated by dividing the output current by the largest output current and then multiplying by a factor. The calculation once again in the overview appeared as follows:

- μ_i: difference between stages of a consumer in amperes (if more than two stages, the average);
- *fi*: punishment of the consumer with the largest μ;
- k_i: weighting for the punishment of the deviation between target and actual value of the i-th consumer.

Thus, the individual cost factors k_i were calculated as follows:

$$k_i = \frac{\mu_i}{\mu_g} * \beta \tag{10}$$

The average differences and cost factors for the different comfort consumers are shown in Table 3.

Table 3. Average differences and cost factors for the electrical consumers ($\beta = -3$).

Electrical Consumer	μ	k
USB	0.77	−0.1
Interior fan	6.58	−0.85
Side mirror heating	2.36	−0.31
Rear window heating	23.09	−3.00
Steering wheel heating	9.43	−1.23
Seat heating driver	12.91	−1.68
Seat heating passenger	15.61	−2.03
Cup holder	2.70	−0.35

To calculate the instantaneous penalty, the deviations from the target statuses of the consumers were multiplied by the respective k. Here, $consumer_{t,i}$ denotes the target value and $consumer_{a,i}$ the actual value of the i-th consumer. Since it was not clear beforehand which number was larger, the subtraction was carried out as follows:

$$p_{consumer,i}(consumer_{t,i}, consumer_{a,i}, k) = |consumer_{t,i} - consumer_{a,i}| * k_i \tag{11}$$

For the loss function with respect to the battery voltage, the difference between a threshold voltage and the actual value was required: $x = V_{actual} - V_{threshold}$. Thus, the penalty was calculated with the following case distinction:

$$p_{voltage}(x) = \begin{cases} 0, & if\ x > 0 \\ 0, & if\ x = 0 \\ -10 - 10^x, & if\ x < 0 \end{cases} \tag{12}$$

The graphical representation of the loss function with respect to the battery voltage is shown in Figure 5.

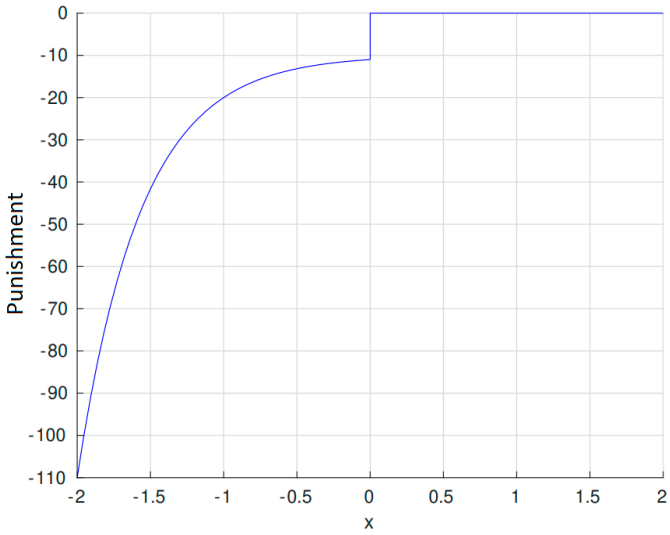


Figure 5. Penalty for the deviations from the voltage target value.

The following applied to the state of charge (SoC) (value range 0 to 1):

$$r_{SOC} = SoC \tag{13}$$

Accordingly, the objective function to be maximized was composed of the individual reward and cost functions (denoted by r and p):

$$f(r,p) = a * r_{SoC} + b * p_{voltage}(x) + c * \sum_{i}^{n} p_{consumer,i}(consumer_{t,i}, consumer_{a,i}, k_i) \tag{14}$$

The objective function set up in Equation (14) formalized the previously mentioned trade-off process between the conservation of the battery and the availability of comfort functions. Although different terms were used for positive and negative values, the objective function was to be maximized. The observations obtained by the respective agent were as follows:

- The actual battery voltage;
- Deviation between target voltage and actual voltage;
- Output current of the high-power consumers;
- SoC;
- All statuses of consumers.

4.3. Training

Since the voltage drops in the case of steering and braking were brief and the penalty via the cost function would hardly be significant, the agents were trained with high-power consumers that were switched on for a longer period of time. Specifically, a dummy high-power load of 600, 800 or 1000 watts was permanently activated starting at second 200 (Figure 6). Here, we made use of the fact that RL agents could be applied to random driving cycles and, thus, trained agents could be used for the original situation from Figure 7. The objective function from Equation (14) was used with a = 3, b = 1 and c = 1.

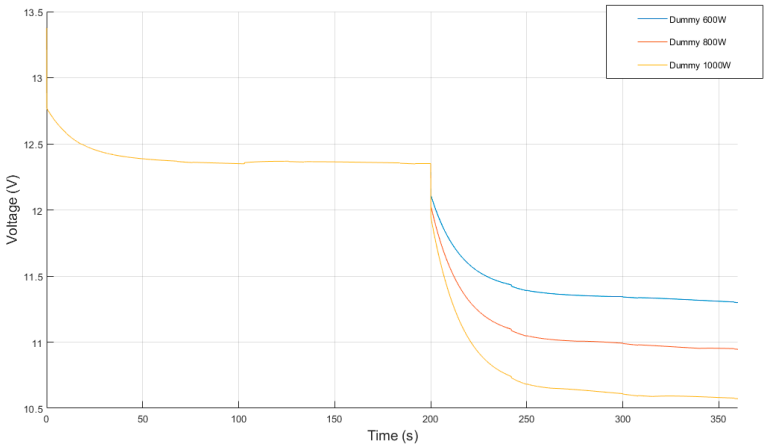
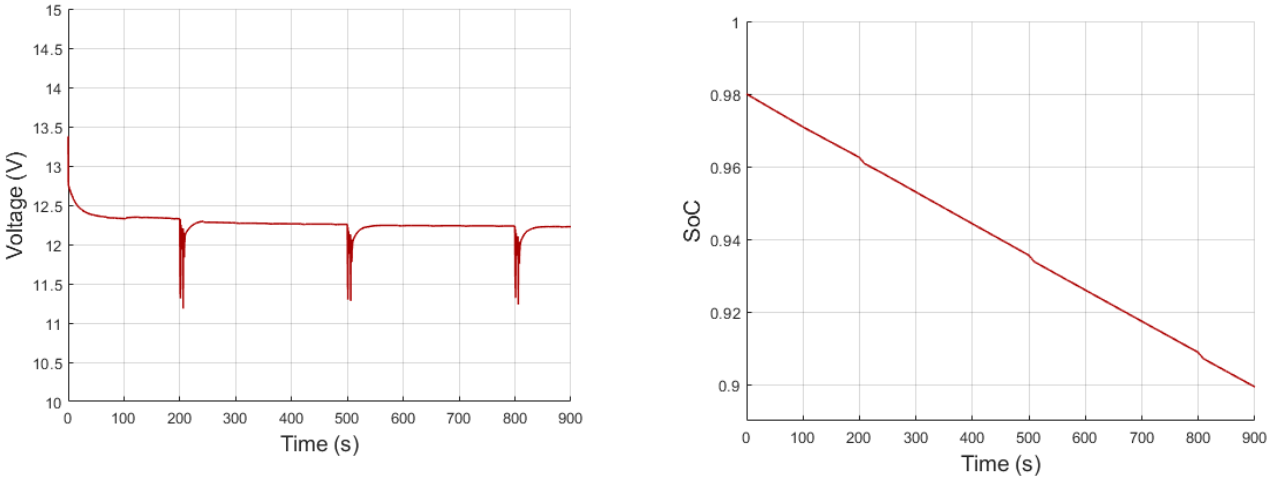


Figure 6. Battery voltage curve with permanent dummy high-power loads.

The scenarios shown in Figure 6 formed the basis for the training runs. The increasingly strong voltage dips were intended to encourage the respective agent to downgrade several consumers. In the present setup, all comfort consumers were activated to the highest level. Furthermore, the agent was not allowed to act freely at any time, but only to intervene when the system was in a critical state, which was defined by the battery voltage.

To test the agents after the training, high-power consumers (heavy steering and braking) were turned on at three time points for 10 s each. The remaining comfort consumers were again all at the highest level. As can be seen in Figure 7, this led in each case to a visible drop in battery voltage. This was also noticeable in the course of the SoC.



(**a**) (**b**)

Figure 7. (**a**) Battery voltage; (**b**) SoC with high-power consumers.

In the area of reinforcement learning based on neural networks to approximate Q-functions, one possible architecture is to use system state features as well as the features of the chosen actions as input. The output is only one neuron representing the Q-function value of the system state and the chosen action. Here, the Q-function values of all actions had to be obtained with serial computations. In this paper, this architecture was chosen for the neural networks. The input of the neural network was chosen to include the sate s of the 12-volt electrical system s = (*battery voltage, deviation between target voltage and actual voltage, current of the high-power consumers, SoC, status of all consumers*) and the chosen action. The number of hidden neurons was intended to be small. For this reason, two hidden layers with 50 neurons each and a rectified linear activation function (ReLU) were used.

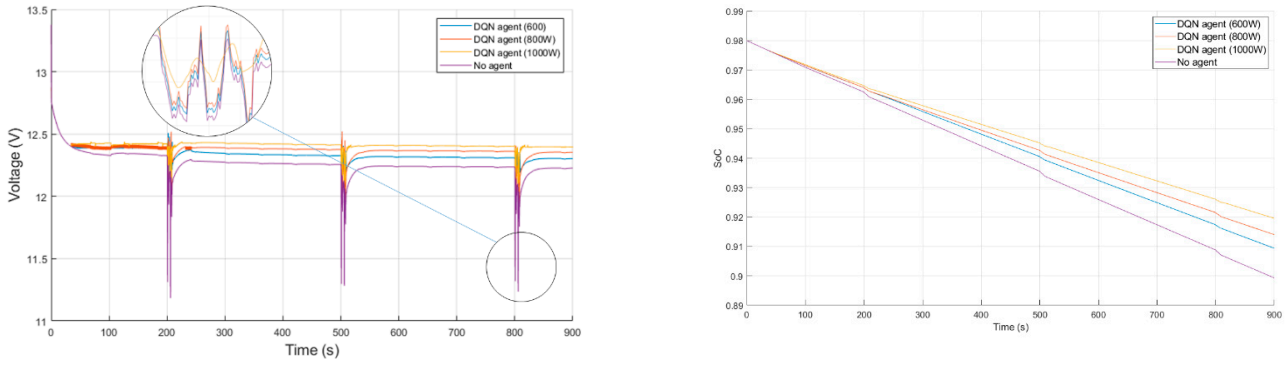
4.4. Results

All three RL agents presented were trained with the dummy high-power consumers and then validated to the profile presented in Figure 7. The tests were always started with an SoC of 98%. After running through the profile of Figure 7, the final SoC was looked at and a conclusion was drawn based on the result.

Figures 8–10 show the respective agents that were applied to the short-term activation of high-power consumers and trained to the permanent activation of dummy high-power consumers. The numbers in parentheses in the legend indicate the power at which the dummy high-power consumer was activated to train the respective agent. In all three agents, it can be seen that training with the highest dummy high-power load gave the best results. The decrease in the SoC at 200s, 500s and 800s can be attributed to turning on the high-power consumers.

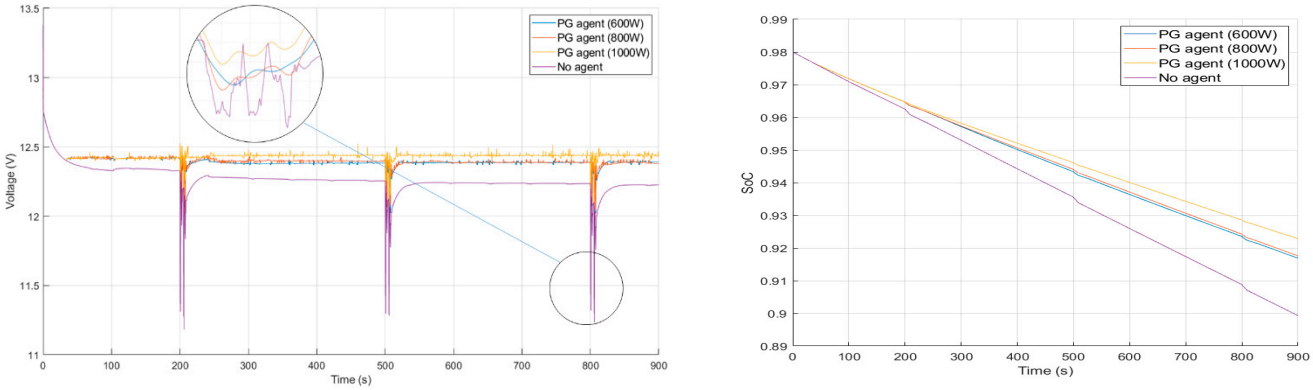
With the best DQN, the battery was discharged by 6.1% at the end of the cycle to 91.9%. In the case of the PG agent, the SoC was 92.2% at the end. Considering the AC agent, the battery was discharged by 5.7%. Thus, the SoC at the end of the driving profile was 92.3%.

Although the sharp drop in battery voltage could be reduced, the many changes in voltage values within a short time can be attributed to frequent state oscillations in the comfort consumers. This effect could be observed for all agents regardless of the scenario, except for the DQN agents, which were trained to 600 W and 1000 W consumers. Therefore, for efficiency reasons, only the DQN agent trained with the 1000 W high-power load was considered in the further course of this work.



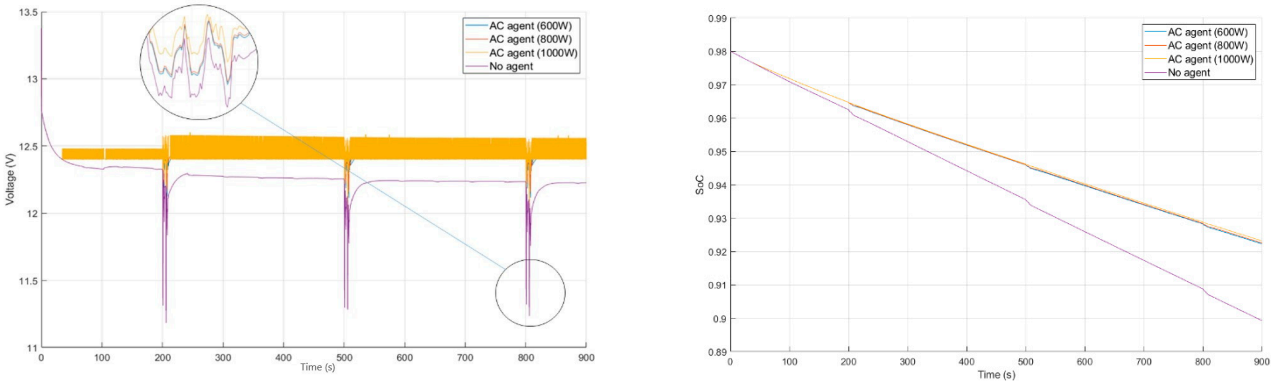
(**a**) (**b**)

Figure 8. (**a**) Battery voltage; (**b**) SoC of the trained DQN agents.



(**a**) (**b**)

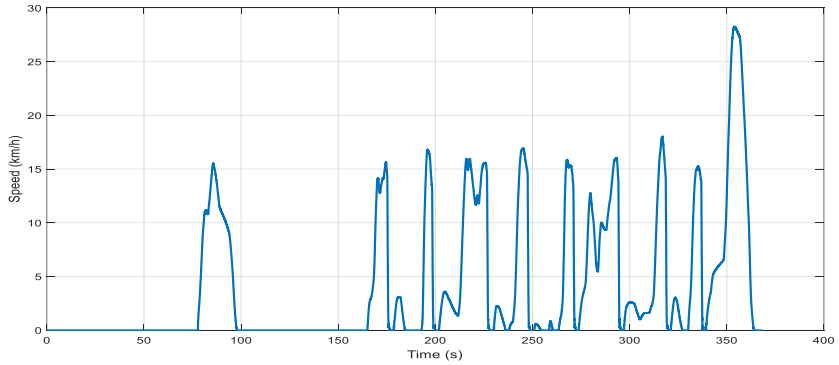
Figure 9. (**a**) Battery voltage; (**b**) SoC of the trained PG agents.



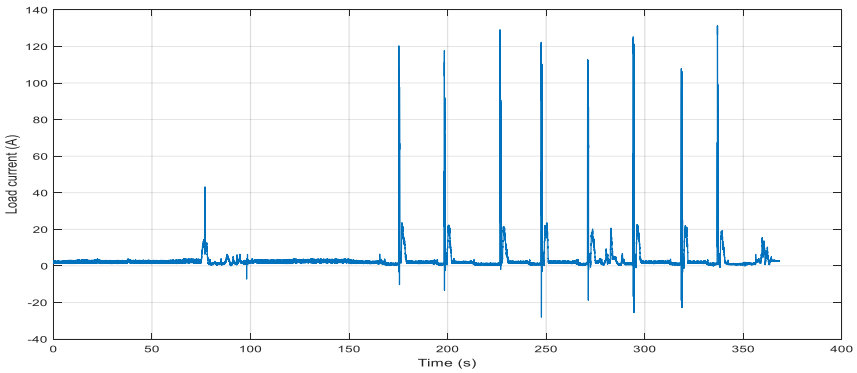
(**a**) (**b**)

Figure 10. (**a**) Battery voltage; (**b**) SoC of the trained AC agents.

Figure 11a shows a speed profile and the high-power consumer profile from an actual measurement. In order to not fully utilize the generator, lower speeds were used. When reaching a certain speed (mostly 15 km/h), a sudden braking and steering maneuver was performed. This resulted in very high currents for a short time in the 12-volt electrical system, which can be seen in Figure 11b. Thus, a worst-case scenario was generated.



(**a**)



(**b**)

Figure 11. (**a**) Speed curve of the validation cycle; (**b**) high-power consumer current of the validation cycle.

Figure 12 presents the SoC comparison after the test cycle. The initial SoC was 83%. It can be clearly seen that in the strong steering and braking phases, the rule-based method adopted a different strategy than that of the DQN agent. In the RB method, a posture of the current SoC was requested for a certain time, while the DQN agent tried to continue to keep the SoC as high as possible. This was, thus, reflected in the result. The final SoC of the agent was at 93%, while that of the RB method ended at 89%.

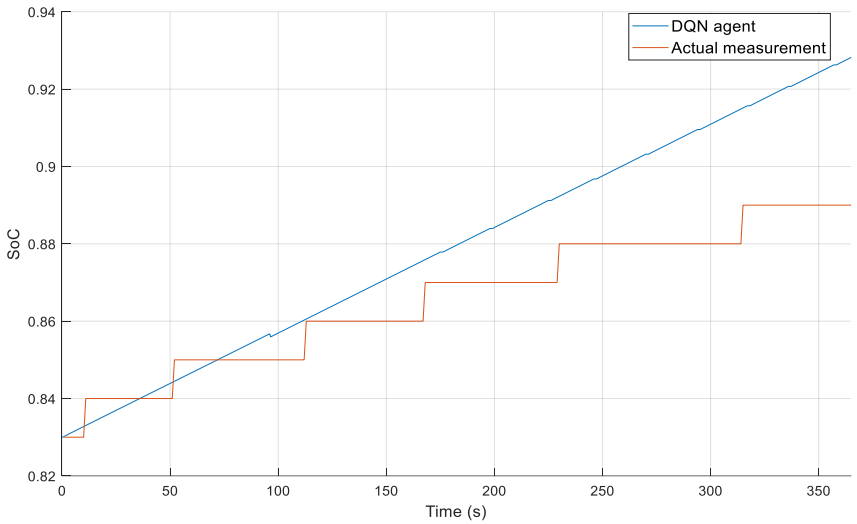
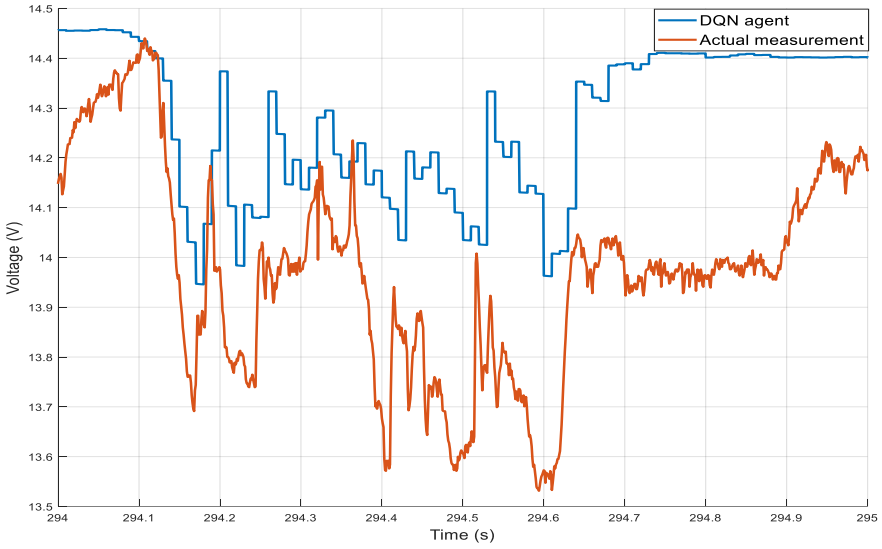
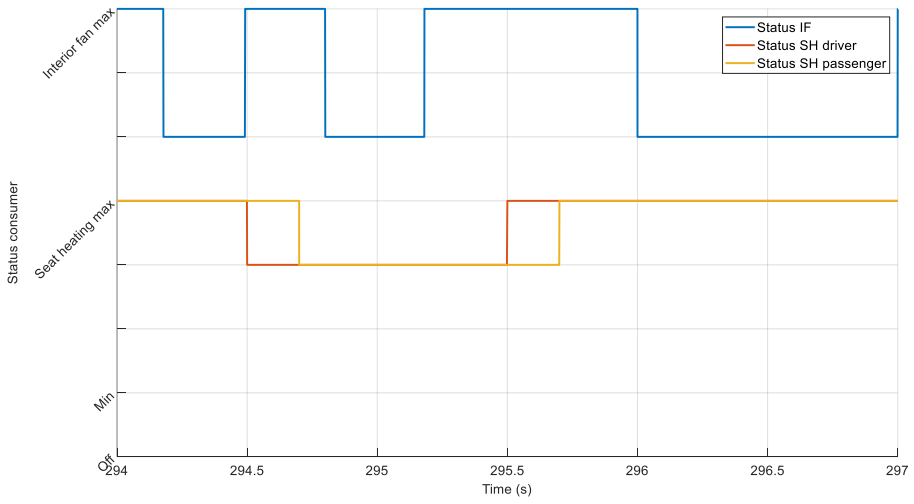


Figure 12. SoC comparison from DQN agent and the actual measurement.

To additionally check the voltage stability, a steering and braking maneuver was observed in more detail. Figure 13 shows one of these maneuvers. In Figure 13a, we can see the voltage drop in the difference. Here, we can clearly see that the agent´s measures lowered the voltage drop. At time t = 294.59 s, the minimum voltage was reduced from 13.53 V to 14.13 V. These measures are shown by some consumer statuses in Figure 13b. It can be seen that in the critical situation, the agent adjusted the status of the comfort consumers, thus, preventing the voltage from dropping further. To make this as unnoticeable as possible to the passengers of the vehicle, the degraded comfort consumers were set back to the target states after a short time.



(**a**)



(**b**)

Figure 13. Comparison of results from simulation to actual measurement. (**a**) Comparison of voltage drops; (**b**) statuses of interior fan, seat heating driver and seat heating passenger controlled by the agent.

5. Conclusions

In this paper, the current state of research regarding the use of an DRL process for an EEMS in the automotive field was discussed. Consequently, the basis for the experiments conducted on the use of an DRL process for an EEMS was derived by analyzing the relevant literature. In the experiments conducted, it was shown that fundamental DRL agents can be used to perform targeted load disconnections to conserve the battery and maintain voltage stability. The main advantage of the DRL-based method was its flexibility to adapt to changing environments and requirements.

Three fundamentally different DRL architectures were applied to the same scenarios with different parameters. Here, the DQN agent proved to be the most effective variant. When investigating voltage stability in the 12-volt electrical system, the DQN agent was able to keep the voltage reduction in critical situations lower than the rule-based variant by means of intelligent coordination of the electrical loads. Based on the conducted experiments, it was shown that DRL-based methods for the optimized coordination of current flows within a 12-volt electrical system offered themselves as alternatives to previous rule-based systems. Concluding measurements showed that the DRL-based method was able to reduce the voltage drops by 0.6 V at critical positions in the 12-volt electrical system during a peak power scenario.

For future work, the simultaneous use of different agents should be investigated. Additionally, the action space of the agent should be extended by further consumers. In addition, more varied driving cycles should better represent the comparison of the DRL-based EEMS to the RB method.

Author Contributions: Conceptualization and validation, Ö.T. and D.J.; methodology, software and investigation, Ö.T.; review and editing, Ö.T., D.J., R.K. and A.T.; supervision, R.K. and A.T.; funding acquisition, A.T. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

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

Abbreviations

The following abbreviations are used in this manuscript:

AC	Actor–critic
DQL	Deep Q-learning
DQN	Deep Q-network
DRL	Deep reinforcement learning
EEMS	Electrical energy management system
EPS	Electric power steering
ESC	Electronic stability control
HEV	Hybrid electric vehicle
ICE	Internal combustion engine
MDP	Markov decision process
NN	Neural network
PG	Policy gradient
QL	Q-learning
RB	Rule-based
ReLU	Rectified linear unit
RL	Reinforcement learning
SoC	State of charge
USB	Universal serial bus

Nomenclature

The following nomenclature is used in this manuscript:

ε	Probability of taking a random action
α, $\alpha^{\theta}, \alpha^{w}$	Step-size parameters
γ	Discount factor
s	State
a	Action
r	Reward
ω	Observation
S	State space
A	Action space
Ω	Observation space
R	Reward function
t	Discrete time step
a_t	Action at time t
s_t	State at time t
ω_t	Observation at time t
r_t	Reward at time t
π	Policy
G_t	Return following time t
θ	Parameter vector of target policy
$J(\theta)$	Performance measure for the policy

References

1. Reif, K. *Sensoren im Kraftfahrzeug*, 2nd ed.; Vieweg+Teubner Verlag: Wiesbaden, Germany, 2012.
2. Fürst, S.; Scharnhorst, T.; Brabetz, L.; Beck, M.; Lahmeyer, R.; Krieger, O.; Kasties, G.; Pfaff, W.; Lachmayer, R.; Abel, H.-B.; et al. *Digitalisierung /Elektrik/Elektronik/Software*; Vieweg Handbuch Kraftfahrzeugtechnik: Wiesbaden, Germany, 2021; pp. 861–1008.
3. Schäuffele, J.; Zurawka, T. *Automotive Software Engineering*; Springer Fachmedien Wiesbaden: Wiesbaden, Germany, 2016.
4. Polenov, D.; Probstle, H.; Brosse, A.; Domorazek, G.; Lutz, J. Integration of supercapacitors as transient energy buffer in automotive power nets. In Proceedings of the European Conference on Power Electronics and Applications, Aalborg, Denmark, 2–7 September 2007; pp. 1–10.
5. Ruf, F. Design and Topology Optimization of Automotive Power Nets in Respect to Voltage Stability. Ph.D. Thesis, Technische Universität München, München, Germany, 2015.
6. Reif, K. *Batterien, Bordnetze und Vernetzung*; Vieweg+Teubner Verlag: Wiesbaden, Germany, 2010.
7. Kohler, T.P.; Wagner, T.; Thanheiser, A.; Bertram, C.; Bücherl, D.; Herzog, H.-G.; Fröschl, J.; Gehring, R. Experimental Investigation on Voltage Stability in Vehicle Power Nets for Power Distribution Management. In Proceedings of the 2010 IEEE Vehicle Power and Propulsion Conference, Lille, France, 1–3 September 2010.
8. Ostadian, R.; Ramoul, J.; Biswas, A.; Emadi, A. Intelligent Energy Management Systems for Electrified Vehicles: Current Status, Challenges, and Emerging Trends. *IEEE Open J. Veh. Technol.* **2020**, *1*, 279–295. [CrossRef]
9. Tang, X.; Chen, J.; Liu, T.; Qin, Y.; Cao, D. Distributed Deep Reinforcement Learning-Based Energy and Emission Management Strategy for Hybrid Electric Vehicles. *IEEE Trans. Veh. Technol.* **2021**, *70*, 9922–9934. [CrossRef]
10. Li, Y.; He, H.; Peng, J.; Wang, H. Deep Reinforcement Learning-Based Energy Management for a Series Hybrid Electric Vehicle Enabled by History Cumulative Trip Information. *IEEE Trans. Veh. Technol.* **2019**, *68*, 7416–7430. [CrossRef]
11. Zhang, F.; Wang, L.; Coskun, S.; Pang, H.; Cui, Y.; Xi, J. Energy Management Strategies for Hybrid Electric Vehicles: Review, Classification, Comparison, and Outlook. *Energies* **2020**, *13*, 3352. [CrossRef]
12. Hu, Y.; Li, W.; Xu, K.; Zahid, T.; Qin, F.; Li, C. Energy Management Strategy for a Hybrid Electric Vehicle Based on Deep Reinforcement Learning. *Appl. Sci.* **2018**, *8*, 187. [CrossRef]
13. Torreglosa, J.P.; Garcia-Triviño, P.; Vera, D.; López-García, D.A. Analyzing the Improvements of Energy Management Systems for Hybrid Electric Vehicles Using a Systematic Literature Review: How Far Are These Controls from Rule-Based Controls Used in Commercial Vehicles? *Appl. Sci.* **2020**, *10*, 8744. [CrossRef]
14. Liu, T.; Hu, X.; Hu, W.; Zou, Y. A Heuristic Planning Reinforcement Learning-Based Energy Management for Power-Split Plug-in Hybrid Electric Vehicles. *IEEE Trans. Ind. Inf.* **2019**, *15*, 6436–6445. [CrossRef]
15. Reif, K. *Automobilelektronik*; Springer Fachmedien Wiesbaden: Wiesbaden, Germany, 2014.
16. Borgeest, K. *Elektronik in der Fahrzeugtechnik*; Springer Fachmedien Wiesbaden: Wiesbaden, Germany, 2014.
17. Fabis, R. Contribution to Energy Management in Vehicles. Ph.D. Thesis, Technische Universität Berlin, Berlin, Germany, 2006.
18. Hosseini, S.M.; Majdabadi, M.M.; Azad, N.L.; Wen, J.Z.; Kothandaraman Raghavan, A. Intelligent Energy Management of Vehicular Solar Idle Reduction Systems with Reinforcement Learning. In Proceedings of the 2018 IEEE Vehicle Power and Propulsion Conference (VPPC), Chicago, IL, USA, 27–30 August 2018; pp. 1–6.

19. Abdelhedi, R.; Lahyani, A.; Ammari, A.C.; Sari, A.; Venet, P. Reinforcement learning-based power sharing between batteries and supercapacitors in electric vehicles. In Proceedings of the 2018 IEEE International Conference on Industrial Technology (ICIT), Lyon, France, 19–22 February 2018; pp. 2072–2077.
20. Jennings, N.; Jennings, N.R.; Wooldridge, M.J. (Eds.) *Agent Technology: Foundations, Applications, and Markets*; Springer: Berlin/Heidelberg, Germany, 1998.
21. Sutton, R.S.; Barto, A. *Reinforcement Learning: An. Introduction*, 2nd ed.; The MIT Press: Cambridge, MA, USA; London, UK, 2018.
22. Zai, A.; Brown, B. *Einstieg in Deep Reinforcement Learning: KI-Agenten mit Python und PyTorch programmieren*; Carl Hanser Verlag GmbH Co KG: Munich, Germany, 2020.
23. Köthe, U. Tiefe Netze. Von Maschinen lernen. *Ruperto Carola* **2020**, *16*, 76–85.
24. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA; London, UK, 2016.
25. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [CrossRef]
26. Schmidhuber, J. Deep learning in neural networks: An overview. *Neural Netw.* **2015**, *61*, 85–117. [CrossRef]
27. Bellemare, M.G.; Naddaf, Y.; Veness, J.; Bowling, M. The Arcade Learning Environment: An Evaluation Platform for General Agents. *J. Artif. Intell. Res.* **2013**, *47*, 253–279. [CrossRef]
28. Munos, R.; Moore, A. Variable Resolution Discretization in Optimal Control. *Mach. Learn.* **2002**, *49*, 291–323. [CrossRef]
29. Brown, N.; Sandholm, T. Superhuman AI for heads-up no-limit poker: Libratus beats top professionals. *Science* **2018**, *359*, 418–424. [CrossRef] [PubMed]
30. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef]
31. Moravčík, M.; Schmid, M.; Burch, N.; Lisý, V.; Morrill, D.; Bard, N.; Davis, T.; Waugh, K.; Johanson, M.; Bowling, M. DeepStack: Expert-level artificial intelligence in heads-up no-limit poker. *Science* **2017**, *356*, 508–513. [CrossRef] [PubMed]
32. Silver, D.; Huang, A.; Maddison, C.J.; Guez, A.; Sifre, L.; van den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. Mastering the game of Go with deep neural networks and tree search. *Nature* **2016**, *529*, 484–489. [CrossRef] [PubMed]
33. François-Lavet, V.; Henderson, P.; Islam, R.; Bellemare, M.G.; Pineau, J. An Introduction to Deep Reinforcement Learning. *FNT Mach. Learn.* **2018**, *11*, 219–354. [CrossRef]
34. Gosavi, A. *A Tutorial for Reinforcement Learning*; Missouri University of Science and Technology: Rolla, MO, USA, 2019.
35. Beutelspacher, A. *Mathe-Basics zum Studienbeginn*; Springer Fachmedien Wiesbaden: Wiesbaden, Germany, 2016.
36. Belousov, B.; Abdulsamad, H.; Klink, P.; Parisi, S.; Peters, J. *Reinforcement Learning Algorithms: Analysis and Applications*; Springer: Cham, Switzerland, 2021.
37. Peters, J. Policy gradient methods. *Scholarpedia* **2010**, *5*, 3698. [CrossRef]
38. Heidrich-Meisner, V.; Lauer, M.; Igel, C.; Riedmiller, M. Reinforcement Learning in a Nutshell. In Proceedings of the ESANN 2007, 15th European Symposium on Artificial Neural Networks, Brügge, Belgien, 25–27 April 2007.
39. Arulkumaran, K.; Deisenroth, M.P.; Brundage, M.; Bharath, A.A. Deep Reinforcement Learning: A Brief Survey. *IEEE Signal Process. Mag.* **2017**, *34*, 26–38. [CrossRef]
40. Konda, V.R.; Tsitsiklis, J.N. Actor-Critic Algorithms. *Adv. Neural Inf. Processing Syst.* **1999**, *12*, 1008–1014.
41. Mismar, F.B.; Choi, J.; Evans, B.L. A Framework for Automated Cellular Network Tuning With Reinforcement Learning. *IEEE Trans. Commun.* **2019**, *67*, 7152–7167. [CrossRef]
42. The MathWorks. Deep Q-Network Agents. Available online: https://de.mathworks.com/help/reinforcement-learning/ug/dqn-agents.html (accessed on 6 February 2022).
43. Al-dayaa, H.; Megherbi, D.B. A Fast Reinforcement Learning Technique via Multiple Lookahead Levels. In Proceedings of the 2006 International Conference on Machine Learning; Models, Technologies & Applications, MLMTA, LasVegas, NE, USA, 26–29 June 2006; pp. 196–202.
44. Williams, R.J.; Peng, J. Function Optimization using Connectionist Reinforcement Learning Algorithms. *Connect. Sci.* **2007**, *3*, 241–268. [CrossRef]