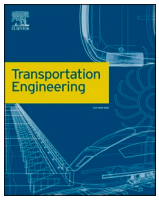
Development of a deep Q-learning energy management system for a hybrid electric vehicle

Transportation Engineering 16 (2024) 100241

Contents lists available at ScienceDirect

Transportation Engineering

journal homepage: www.sciencedirect.com/journal/transportation-engineering



Full Length Article

Development of a deep *Q*-learning energy management system for a hybrid electric vehicle

Luigi Tresca, Luca Pulvirenti, Luciano Rolando *, Federico Millo

Politecnico di Torino, C.so Duca degli Abruzzi, 24, 10129 Turin, TO, Italy

ARTICLE INFO

Keywords:
Hybrid electric vehicle
Energy management system
Artificial intelligence
Reinforcement learning

ABSTRACT

In recent years, Machine Learning (ML) techniques have gained increasing popularity in several fields thanks to their ability to find hidden and complex relationships between data. Their capabilities for solving complex optimization tasks have made them extremely attractive also for the design of the Energy Management System (EMS) of electrified vehicles. Among the plethora of existing techniques, Reinforcement Learning (RL) algorithms have unprecedented potential since they can self-learn by directly interacting with the external environment through a trial-and-error procedure. In this paper, a Deep *Q*-Learning (DQL) agent, which exploits Deep Neural Networks (DNNs) to map the state-action pair to its value, was trained to reduce the CO_2 emissions of a state-of-the-art diesel Plug-in Hybrid Electric Vehicle (PHEV) available on the European market. The proposed methodology was tested on a virtual test rig of the investigated vehicle while operating on a charge-sustaining logic. A sensitivity analysis was performed on the reward to test the capabilities of different penalty functions to improve the fuel economy while guaranteeing the battery charge sustainability. The potential of the proposed control strategy was firstly assessed on the Worldwide harmonized Light-duty vehicles Test Cycle (WLTC) and benchmarked against a Dynamic Programming (DP) optimization to evaluate each reward. Then the best agent was tested on a wide range of type-approval and Read Driving Emission (RDE) scenarios. The results show that the best-performing agent can reach performance close to the DP reference, with a limited gap (7 %) in terms of CO_2 emissions.

Abbreviations

BEV	battery electric vehicle
BSFC	brake specific fuel consumption
DDPG	deep deterministic policy gradient
DNN	deep neural network
DP	dynamic programming
DQL	deep *Q*-learning
DDQL	double deep *Q*-learning
ECMS	equivalent consumption minimization strategy
ECU	electronic control unit
EM	electric machine
EMS	energy management system
FTP	federal test procedure
GHG	greenhouse gas
HEV	hybrid electric vehicle
ICE	internal combustion engine
LB	learning based
MC	Monte Carlo
ML	machine learning
MPC	model predictive control
NN	neural network
PHEV	plug-in hybrid electric vehicle
PMP	Pontryagin's minimum principle
QL	*Q*-learning
RB	rule based
RDE	real driving emission
RESS	rechargeable energy storage system
RL	reinforcement learning
RNN	recurrent neural network
SAC	soft actor-critic
SoC	state of charge
TD	temporal difference
WLTC	worldwide harmonized light-duty vehicles test cycle

* Corresponding author.
E-mail address: luciano.rolando@polito.it (L. Rolando).

https://doi.org/10.1016/j.treng.2024.100241
Received 16 October 2023; Received in revised form 20 February 2024; Accepted 3 March 2024
Available online 7 March 2024

1. Introduction

Reducing the carbon footprint of the transportation sector, which accounts for roughly 35 % of the total worldwide energy consumption [1], is imperative for reaching carbon neutrality by 2050, as planned by the EU "Fit for 55 package" [2]. Conventional vehicles, propelled only by an Internal Combustion Engine (ICE), cannot reach the CO_2 emission targets set by the upcoming regulations. The Greenhouse Gas (GHG) emissions of current vehicle fleets can be strongly reduced by the synergistic exploitation of eco-driving algorithms [3], that optimize the vehicle speed in a connected environment, along with electrified mobility solutions. In this context, powertrain hybridization can represent, at least in the short term, a viable solution to improve powertrain efficiency, while mitigating the current disadvantages of Battery Electric Vehicles (BEVs), such as limited range, long recharging time, and lack of an adequate infrastructure [4].

However, the introduction of an auxiliary energy source on board must be properly managed to fully exploit the benefits of powertrain electrification, necessitating the redesign of the vehicle control hierarchy with the introduction of an additional layer, called the Energy Management System (EMS). In a hybrid powertrain, the EMS role is to control the powertrain operating mode and the power split between the ICE and the Electric Machines (EMs) [5]. Various approaches can be used to design the EMS, broadly divided between online implementable and offline optimization methods [6]. A comprehensive review of current state-of-the-art strategies, analyzing more than 250 EMS-related publications, can be found in [7].

Among online approaches, Rule-Based (RB) strategies are the most common, relying on pre-tuned rules and lookup tables that can be easily implemented online but may typically achieve results quite far from optimality. On the other hand, among the offline optimization methods, Dynamic Programming (DP) [8] can provide the global optimum in terms of fuel savings but requires high computational costs and the a-priori knowledge of the complete driving mission profile. For these reasons, DP can only be used as a benchmark. A compromise solution is represented by local optimization strategies, e.g., the Equivalent Consumption Minimization Strategy (ECMS) [9], or Pontryagin's Minimum Principle (PMP) [10], that instantaneously minimize fuel consumption, providing sub-optimal results while being implementable on a vehicle Electronic Control Unit (ECU) [11]. The current trend for these decision strategies is toward real-time implementable solutions.

Another option for designing the EMS of HEVs is Model Predictive Control (MPC) [12], which enlarges the optimization horizon from a local to a short-term perspective. MPC can easily tackle constraints on state and control variables, while being low demanding in terms of computational requirements, thus online implementable. However, its main drawback is its dependency on accurate prediction over the optimization scenario. Examples of MPC applications on real vehicles considering aspects like fuel economy and battery lifespan are found in Wang et al. [13] and Williams [14].

Recently, also Learning-Based (LB) approaches have become widely adopted in the automotive sector [7]. Relying on Machine Learning (ML) techniques, these approaches are extremely promising thanks to their ability to uncover hidden and complex relationships in data characterizing the modeled problem. ML models are particularly suitable for applications that involve highly non-linear relationships [15], where an RB description might become complex and not particularly accurate. As an example, in Millo et al. [16] the Authors developed an ML-based EMS by training two deep Recurrent Neural Networks (RNNs) [17] to emulate the optimal control provided by DP across a wide range of driving scenarios.

Among the plethora of ML techniques, Reinforcement Learning (RL) algorithms [18] are notably promising if employed for planning and optimization tasks, since they can self-learn by directly interacting with the external environment (i.e., vehicle model, driving cycle, etc.) through a trial-and-error process. The application of RL in the energy management of a Hybrid Electric Vehicle (HEV) consists in gradually learning the optimal strategy aimed at minimizing a performance index by penalizing fuel-intensive operations and rewarding fuel-efficient ones.

On the other hand, the main drawback of RL techniques lies in a tricky and time-consuming training process since it depends on the tuning of various parameters, called hyperparameters. Particularly crucial is the proper definition of the reward function, the core element of the RL agent. Among the RL agents, *Q*-Learning (QL) [19] stands out as a prominent off-policy tabular algorithm, serving as a benchmark for addressing control problems in discretized environments. While the literature presents other techniques capable of dealing with different types of environments (discrete-continuous or completely continuous), the most promising ones include the following: Deep *Q*-Learning (DQL) [20], which exploits Deep Neural Networks (DNNs) [21] to approximate the value function; Deep Deterministic Policy Agent (DDPG) [22], based on two distinct DNNs to approximate the policy and the value functions; and Soft Actor-Critic (SAC) [23], which uses a stochastic policy approximator for dealing with continuous environments. Nevertheless, despite the increased potentialities, the additional components employed in DDPG and SAC increase the agent's complexity, thereby making QL and DQL agents highly appealing and adopted choices.

As an example, in Xu et al. [24], a QL method was used to optimize the EM torque in a mild-HEV, without considering the battery State of Charge (SoC) as a state. This approach improved the fuel economy of the vehicle by about 9 % if compared to an RB strategy. However, the implemented QL method exploited a tabular-based agent, whose performance highly depends on the number of states and their discretization, suffering the so-called "curse of dimensionality". Therefore, this approach allows only considering simple tables, and, as demonstrated by the parametric study conducted in Xu et al. [25], increasing the number of states and their discretization resolution does not always lead to a fuel economy improvement. Moreover, in these activities, an ECMS-style reward was employed to train the agent. Despite allowing the agent to outperform an RB control strategy, this reward can be highly case-dependent, limiting the agent's applicability to driving cycles different from the training one.

Further developments of the QL method were applied to a series [26] and a parallel HEV [27], where the agent received relevant information about the driving mission by using a transition probability matrix and used a reward function that only took into account the fuel consumption. Another application of tabular *Q*-learning can be found in Musa et al. [28], where also comfort and engine operation requirements are considered other than fuel consumption. Nevertheless, despite their ease of implementation, QL agent performance is limited by the adoption of the tabular approach. The *Q*-value performance can be boosted by using NNs to approximate the value function. For instance, in Zou et al. [29] DQL was employed in the design of the EMS of a series HEV: the agent was trained through a prioritized replay module to speed up the training phase and then tested in a hardware-in-the-loop environment. The reward was designed considering both the cost of fuel and electric energy and adding a term preventing rapid SoC fluctuations. In [30], a DQL-based EMS was applied to a parallel hybrid electric bus continuously penalizing the fuel consumption and the SoC deviation from the target, improving the fuel economy by 5.6 % if compared with conventional QL.

Despite improving the performance, DQL tends to overestimate the *Q*-value during the training phase. To mitigate this concern, Double Deep *Q*-learning (DDQL) employs two neural networks, the online and target networks, to separately handle action selection and value estimation [31]. For instance, in Han et al. [32] a DDQL agent was adopted for a dual-motor driven hybrid electric tracked-vehicle showing better performance than the conventional DQL approach. The adoption of actor-critic agents, such as DDPG and SAC, allows for improving the performance at the expense of an increased level of complexity since they exploit DNNs to approximate both the policy and the value

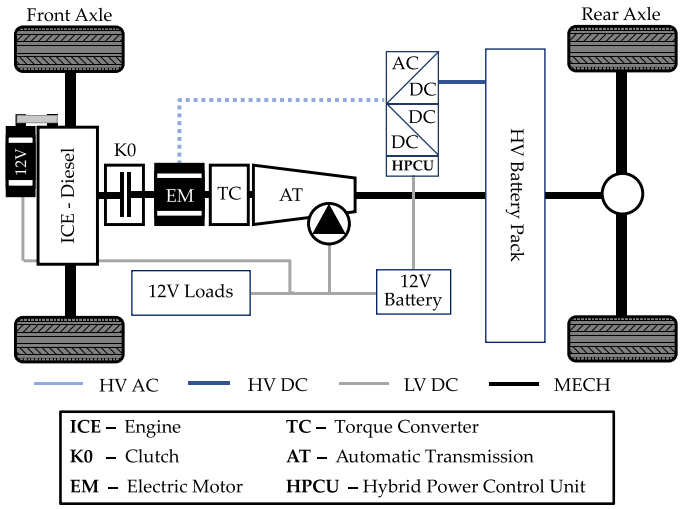


Fig. 1. Powertrain layout: a diesel engine is connected through an auxiliary clutch (K0) to an EM. Both the ICE and the EM are connected, through a torque converter, to the automatic transmission. HV AC, HV DC, and LV DC refer to the electric connections, while MECH refers to the mechanical connections.

functions. As an example, in Lian et al. [33] a DDPG agent was used to solve a multi-objective energy management problem within a large control variable space continuously penalizing the agent with the fuel consumption and the SoC deviation from the target. Instead, in Haarnoja et al. [34], a SAC agent was developed incorporating a penalty term proportional to the SoC deviation, in addition to fuel consumption, only if the SoC was less than the target value. The vehicle energy consumption was reduced by 4.4 % if compared to an ECMS, demonstrating also adaptability to different driving cycles. A SAC agent was also proposed in Huo et al. [35] that optimized not only energy consumption but also fuel cell and battery lifetime awareness.

Nevertheless, few works exploiting RL techniques for the energy management of HEVs have properly addressed the core aspect of this algorithm, which is the reward definition. In the literature, the agent of an RL algorithm is usually trained through a single reward function, but the type of reward deeply affects the performance of the agent [18]. Since the agent aims to maximize the cumulative reward, a badly designed reward may lead to unsatisfactory results, whatever the outcome of the training process. Therefore, special attention should be paid to the reward definition to guide the agent towards a sub-optimal policy. In this context, the purpose of this study is to evaluate the impact of different reward functions on the performance of the EMS of a Plug-in Hybrid Electric Vehicle (PHEV) operating in charge-sustaining mode. Four different reward formulations were proposed to assess their impact on the agent's performance in optimizing energy flows while also ensuring charge sustainability.

For this purpose, a DQL agent was chosen thanks to its ability to represent high-dimensional observations without the increased complexities of actor-critic agents. The DQL was trained on the Worldwide harmonized Light-duty vehicles Test Cycle (WLTC) since it is as representative as possible of real-world driving conditions and comprises different driving patterns, i.e., urban, rural, and highway. The proposed rewards were then compared in terms of vehicle fuel economy and battery charge sustainability at the end of the driving cycle, using the global optimal solution provided by DP the global to benchmark their performance. Furthermore, the evaluation of the reward formulations also considered their training performance, aiming to achieve the quasi-optimal solution as fast as possible while ensuring training stability, i.e., consistent results across both the test and training phases. Finally, the best reward was tested on driving scenarios different from the training one to assess the adaptability of the methodology to other driving cycles.

The paper is organized as follows: after a brief introduction of the case study (Section 2), the formulation of the energy management problem is introduced (Section 3), with a particular focus on DQL and reward functions definition. The performance of the reference reward is shown on the WLTC (Section 4.1), followed by a sensitivity analysis of the behavior of each reward (Section 4.2). Then, the results of the best-performing agent are shown on a wide range of test scenarios (Section 4.3). Finally, the paper summarizes the main findings of the research activity and its possible future developments.

Table 1
Vehicle and powertrain main specification.

Vehicle	
Curb Weight	2060 kg
Power Demand @ 100 km/h	14.9 kW
Configuration	Rear Wheel Drive
Transmission	
Type	9-AT w/ Torque Converter
Engine	
Type	In-line 4 cylinders Turbo Diesel
Displacement	1950 cm^3
Max Power	143 kW @ 3800 rpm
Max Torque	400 Nm @ 1600–2800 rpm
Electric Machine	
Type	PM Synchronous Motor
Max Power/ Max Torque	90 kW @ 2000 rpm / 440 Nm @ 1750 rpm
Max Speed	6000 rpm
High Voltage Battery	
Type	Li-NMC
Rated Voltage	365 V
Capacity	13.5 kWh / 37 Ah

2. Case study

The vehicle under investigation is a state-of-the-art diesel PHEV available in the European market. The powertrain layout is shown in Fig. 1, while Table 1 summarizes the main vehicle and powertrain features. The hybrid powertrain has a P2 architecture, where a Euro 6d-temp 1950 cc diesel engine is integrated with a 90 kW EM. Both the ICE and the EM are connected, through a torque converter and a 9-speed automatic transmission to the rear axle. In previous studies, the vehicle was extensively investigated through an experimental campaign, and a virtual test rig was built and validated against the experimental data [36]. Since this work is focused on the effect of different rewards on the DQL agent performance, a simplified version of the digital twin, that relies on a backward kinematic approach [37], was developed in MATLAB®. Utilizing the simplified vehicle model allowed for speeding up, without compromising on accuracy, the agent's training process, which would have otherwise required a huge computational effort if performed on a forward dynamic model.

3. Energy management

3.1. Problem formulation

Energy management in an HEV can be regarded as an optimal control problem where the goal is usually to minimize fuel consumption. In this case, the cost function J can be defined as:

$$J = \int_{t_0}^{t_f} \dot{m}_f(\pi(S), t)dt \qquad \pi : S \rightarrow A \tag{1}$$

where $\dot{m}_f$ [g/s] is the instantaneous mass flow rate, t is the time variable, and π is the policy function, which maps an action A to every observable state S [38]. Since the cost function must be minimized under a set of both local and global constraints, the energy management problem of an HEV is a constrained and finite-time optimal control problem: the minimization of J is subject to constraints related to the physical

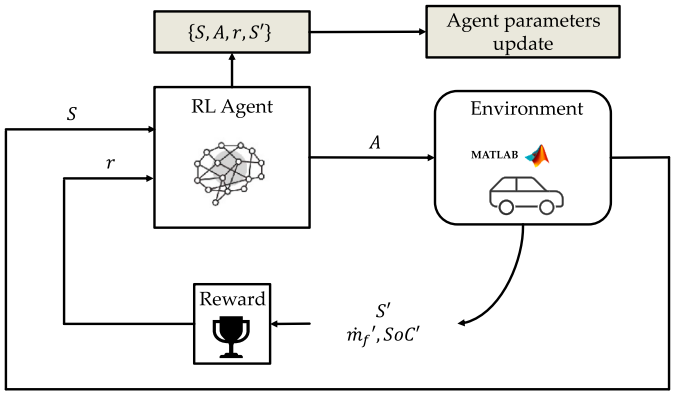


Fig. 2. Schematic representation of the RL algorithm: the RL agent interacts with the external environment and learns through a trial-and-error process.

limitations of the actuators, i.e., ICE and EM, and of the Rechargeable Energy Storage System (RESS), i.e., the vehicle battery, as expressed in the following:

$$\begin{aligned} &P_{ICE,\min}(\omega_{ICE}(t)) \le P_{ICE}(t) \le P_{ICE,\max}(\omega_{ICE}(t)) \\ &P_{EM,\min}(\omega_{EM}(t)) \le P_{EM}(t) \le P_{EM,\max}(\omega_{EM}(t)) \\ &P_{batt,\min}(SoC(t)) \le P_{batt}(t) \le P_{batt,\max}(SoC(t)) \end{aligned} \tag{2}$$

where ω_{ICE} and ω_{EM} are the engine and the electric motor speeds, respectively. Moreover, for an HEV or a PHEV operating in charge-sustaining mode, battery charge sustainability must be ensured, thus the battery SoC must be always contained within prescribed limits as follows:

$$\begin{aligned} &SoC_{\min} \le SoC(t) \le SoC_{\max} \\ &SoC(t_f) = SoC(t_i) \end{aligned} \tag{3}$$

As already discussed in Section 1, the described control problem can be addressed with well-known methods, such as DP, ECMS, PMP, etc. Nevertheless, quite recently, the development of EMSs gained a giant leap towards attaining a real-time global optimum control with the introduction of Reinforcement Learning (RL) [18] as a control strategy since it combines real-time implementation characteristics from local methods and global optimization characteristics from DP. RL is based on the concept that an agent, i.e., the ML-based EMS, can be trained by directly interacting with an external environment through a trial-and-error process.

The operating principles of RL can be better understood from Fig. 2, where a schematic representation of the process involved in the RL training is shown: at each time step, the agent selects an action A depending on the actual policy π which leads to a reward r and a new state $S^{'}$. The goal of any RL algorithm is to exploit the transition information $\{S,A,r,S^{'}\}$, which is also called an experience, to find the optimal policy π_*. A policy is optimal if it maximizes the return g_t, which can be expressed as a weighted sum of the temporal reward as follows:

$$g_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \tag{4}$$

where $\gamma \in [0,1]$ is the discount factor: it weighs the influence of future rewards, leading the agent to a short or long-sighted behavior. The higher the discount factor, the higher the influence of future rewards on the current agent's behavior.

3.2. Deep Q-learning

Different types of agents are available for the RL in the literature, such as policy-based, value-based, or a combination of them [18]. Furthermore, the agent's capability to handle either a discrete or continuous state-action space depends on the type of model employed. For example, tabular-based agents, which may be policy or value-based, can only deal with a discrete state-action space, while agents based on Neural Networks (NNs) allow the introduction of a continuous state space.

Among the RL agents, QL is one of the most popular. It is model-free since no model of the environment's dynamics is required, and value-based since it does not directly update the optimal policy but rather estimates the Q-value function. This function represents a refined estimation of the expected future reward (see Eq. (4)) obtained by taking action A in state S, and following a target policy thereafter. Moreover, QL employs Temporal Difference (TD) learning [18], thus combining elements of both Monte Carlo (MC) methods [39] and DP. As a matter of fact, like MC methods, TD can learn by directly interacting with the environment, without requiring a model of its dynamics. Like DP, TD employs a "bootstrapping" approach [40], i.e., it estimates the Q-value function based on estimates of successor state-action pairs.

The Q-value functions can be approximated using either tables or NNs. In the latter case, the agent, called Deep Q-Learning (DQL), is trained off-policy and exploits two different NNs to enhance training stability. The first NN, referred to as the behavior Q-value function, interacts with the environment and generates the transition *{S,A,r,S'}* (see Fig. 2), while the second NN, referred to as the target Q-value function, is updated with a customized frequency, and actually used by the agent after the training phase. During the training phase, the DQL algorithm initializes the parameters of the NNs randomly, and the agent selects and executes an action A at each time step as follows:

$$A = \begin{cases} \underset{A}{\text{argmax}} Q_b(S,A,\Phi_b) \text{ with probability } 1-\epsilon \\ \text{random action with probability } \epsilon \end{cases} \tag{5}$$

where Q_b is the behavior Q-value function defined by an NN, S is the actual state, A is the set of available actions, Φ_b represents the parameters of the behavior Q-value function, and ϵ is a calibration parameter that sets the exploration level. During the training, the agent selects a random action with a probability ϵ. This exploration strategy is called ϵ-greedy, since the greedy actions (chosen with a probability $1-\epsilon$) are the ones that exploit the current knowledge to maximize the behavior Q-value function, while the ϵ parameter allows the exploration of the state-action space by randomly selecting actions. The higher the ϵ value, the wider the agent explores the state-action space. In the tuning phase, the ϵ parameter must be carefully chosen to balance the exploration/exploitation trade-off. The exploration strategy plays a crucial role since the maximization of the reward is a trial-and-error process and, the exploration of new state-action pairs may lead to an improvement of the current policy while exploitation will only behave optimally given the current knowledge.

In the DQL, analogously to all the RL algorithms (see Fig. 2), the agent receives a reward r and a new state $S^{'}$ depending on the selected action A. The information linked to a transition, i.e., $\{S,A,r,S^{'}\}$, is called an experience. In this case, however, the training is performed off-policy, which means that the last experience is not directly used to update the agent parameters but is stored in an experience buffer. Then, the parameters update is performed by randomly sampling a mini-batch of M experiences $\{S,A,r,S^{'}\}$ from the experience buffer. For each experience in the mini-batch, the action that maximizes the Q-value in the next state conditions $S^{'}$ is computed from the behavior network as follows:

$$A^{'}_{\max} = \underset{A^{'}}{\text{argmax}} Q_b(S^{'}_i, A^{'}, \Phi_b) \tag{6}$$

where $S^{'}$ and $A^{'}$ are the state and the set of actions at time $t+1$, Φ_b represents the parameters of the behavior Q-value function, and i indicates the ith experience into the M experiences mini-batch.

The target Q-value function, instead, is used to generate a target value y_i as follows:

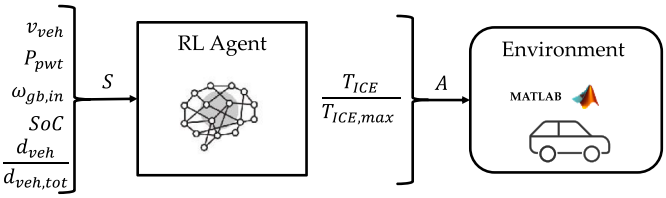


Fig. 3. Schematic representation of the application of the RL algorithm to the case study, with employed states and action.

$$y_i = r_i + \gamma Q_t\left(S_i^{'}, A_{\max}^{'}, \Phi_t\right) \tag{7}$$

where Q_t and Φ_t refer to the target Q-value, which is defined by an NN, γ is the discount factor, and i refers to the ith experience of the mini-batch. For each experience, the parameters of the behavior network are updated to minimize the cost function L:

$$L = \frac{1}{2M}\sum_{i=1}^{M}\left(y_i - Q_b(S_i, A_i, \Phi_b)\right)^2 \tag{8}$$

and a backpropagation algorithm, namely the Adam optimizer [41], is used for training the NNs. Finally, at the end of each episode, i.e., a driving cycle, also the parameters of the target NNs are updated by implementing the optimized ones of the behavior NNs. It should be noted that training the agent off-policy results in a more sample-efficient process as M different experiences are used to update the agent parameters at each time step instead of relying on a single experience. Moreover, the introduction of an additional network, i.e., the target Q-value function, enhances training stability, since the target NNs are updated only at the end of each episode.

3.3. Application to case study

In this work, as schematically shown in Fig. 3, five states are provided to the agent:

- Vehicle speed v_{veh};
- Powertrain power demand P_{pwt};
- Gearbox inlet speed $\omega_{gb,in}$;
- Battery state of charge SoC;
- Traveled distance over the total distance $\frac{d_{veh}}{d_{veh,tot}}$.

The action is, instead, the engine torque T_{ICE} divided by the maximum available torque to avoid unfeasible engine operating conditions.

Differently from DP, RL cannot enforce a constraint on the final value of SoC to guarantee charge sustainability. As already mentioned in the introduction, the design of a suitable reward is crucial to guide the RL agent towards a sub-optimal policy that guarantees charge-sustaining conditions. In this work, four different reward formulations have been proposed to assess their impact on the agent's performance in optimizing energy flows while also ensuring charge sustainability. It is noteworthy that, in line with the modeling approach adopted for implementing the DP, the defined rewards do not consider any drivability constraints. The four reward formulations are briefly outlined below.

- *Case 1*

Since, in charge-sustaining mode, the main goal of the strategy is to minimize fuel consumption while guaranteeing charge sustainability, the first reward function was thought to take into account both the ICE fuel consumption and a penalty if the SoC deviates from the target value. It is formulated as follows:

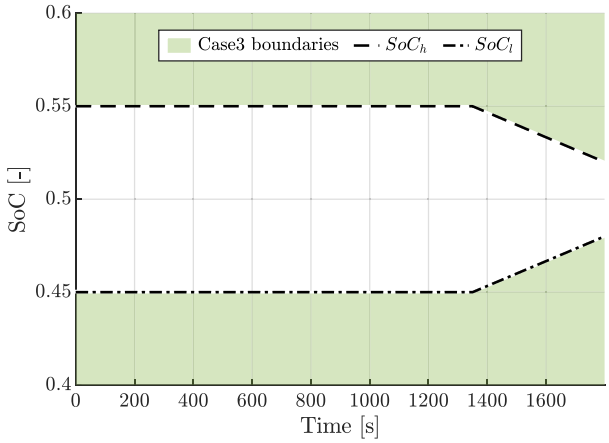


Fig. 4. SoC boundary lines for Case 3 reward.

$$r_{Case1} = k_1 - k_2\dot{m_f}\Delta t - k_3\left(SoC - SoC_{trg}\right)^2 \tag{9}$$

where k_1, k_2 and k_3 are constant which weigh the influence of the relative terms, $\dot{m}_f$ is the fuel rate, SoC and SoC_{trg} are the actual and the target SoC, respectively. This reward uses a continuous penalization on the SoC depending on the distance from the SoC target. Since the goal of the agent is to maximize the reward (see Eq. (6)), while the EMS objective is to minimize fuel consumption, the latter cannot be directly defined as a reward. To overcome this problem, the ICE fuel consumption ($\dot{m_f}\Delta t$) as well as the SoC penalty is changed of sign. Despite not being necessary, a positive or null offset k_1 can be added to the reward to make it positive. Although this type of reward is easier to train, as will be shown in Section 4, it introduces a penalty related to the SoC deviation during the entire simulation. A numerical trade-off between fuel saving and SoC deviation from the target must therefore be considered in the definition of the reward function, thus the constants k_1, k_2, and k_3 must be properly tuned before training.

- *Case 2*

This reward was thought, differently from *Case 1*, to penalize the agent only if the SoC deviates from the target value at the end of the driving cycle. The reward is formulated as follows:

$$r_{Case2} = k_1 - \left(k_2\dot{m}_f + k_3 P_{SoC}\right)\Delta t$$
$$P_{SoC} = \begin{cases} 0 & \text{if } t < t_f \\ \left(SoC - SoC_{trg}\right)^2 & \text{if } t = t_f \end{cases} \tag{10}$$

where k_1, k_2 and k_3 are constant which weigh the influence of the relative terms, P_{SoC} is the factor that penalizes the reward if the SoC deviates from the target, and t_f is the final time of the driving cycle. This reward can theoretically obtain better results in terms of fuel saving if compared to *Case 1*: the trade-off between fuel saving and SoC deviation from the target is no longer present, thus the agent has more freedom to choose the optimal action. However, it makes the training process of the agent more complex: if the agent is penalized only at the end of the driving cycle, there are fewer occurrences in which the SoC penalty is applied if compared to *Case 1*.

- *Case 3*

This case was introduced to merge the ideas of *Case 1* and *Case 2*, as can be seen from the following reward:

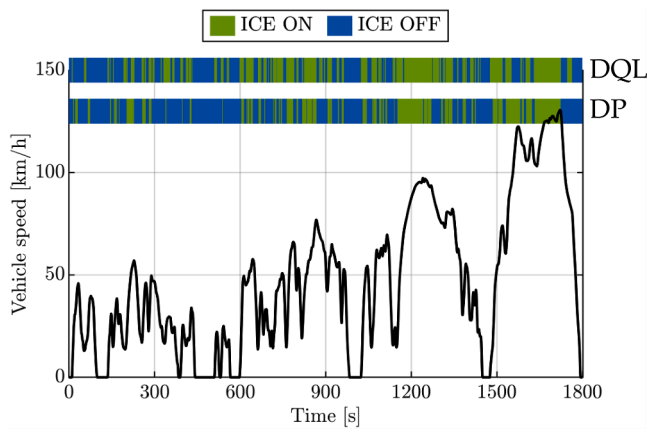


Fig. 5. Operating modes selected by DQL and DP strategies on the WLTC: parallel mode (green) and only-electric mode (blue). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

$$r_{Case3} = k_1 - \left(k_2 \dot{m}_f + k_3 P_{SoC}\right)\Delta t$$
$$P_{SoC} = \begin{cases} \left(SoC - SoC_{trg}\right)^2 & \text{if } SoC < SoC_l \\ 0 & \text{if } SoC_l \le SoC \le SoC_h \\ \left(SoC - SoC_{trg}\right)^2 & \text{if } SoC > SoC_h \end{cases} \quad (11)$$

The two SoC boundary lines, SoC_l and SoC_h, are displayed in Fig. 4. They are conceptually derived from the DP boundary lines method described in Sundström et al. [42] since, between them, the agent is free to choose the optimal action, and no penalty is applied if the SoC deviates from the target. However, if the agent chooses an action that causes the SoC to take a value outside the boundaries, a penalty is applied to the reward depending on the distance between the actual and the target SoC values. Thus, *Case 3* allows the agent to freely choose the action that maximizes the reward as long as the boundary lines are not exceeded.

- *Case 4*

This case was designed by introducing the reward proposed in Xu et al. [24]. It is an ECMS-style reward, expressed as follows:

$$r_{Case4} = k_1 - m_{eq}$$
$$m_{eq} = \left(\dot{m}_f + \dot{m}_{f,batt}\right)\Delta t \quad (12)$$
$$\dot{m}_{f,batt} = s\frac{P_{batt}}{Q_{LHV}}$$

where the equivalent battery fuel consumption ($\dot{m}_{f,batt}$) is computed from the battery power (P_{batt}) and the fuel's lower heating value (Q_{LHV}). The main drawback of this approach is that the equivalence factor s must be properly tuned in advance to obtain the charge sustainability at the end of the cycle. Thus, the *Case 4* reward is case-sensitive: using this reward in an RL framework with different driving scenarios may require more calibration effort.

3.4. Tools and libraries

As mentioned in Section 2, the simplified version of the digital twin, relying on a backward kinematic approach was developed in MATLAB®. The environment, consisting of the digital twin of the vehicle running on standardized cycles was modified to make it compliant with the rlDQNAgent object. For performing the DP optimization, employed for benchmarking the RL algorithm performance, the open-source MATLAB® code developed at ETH-Zurich [43] was used.

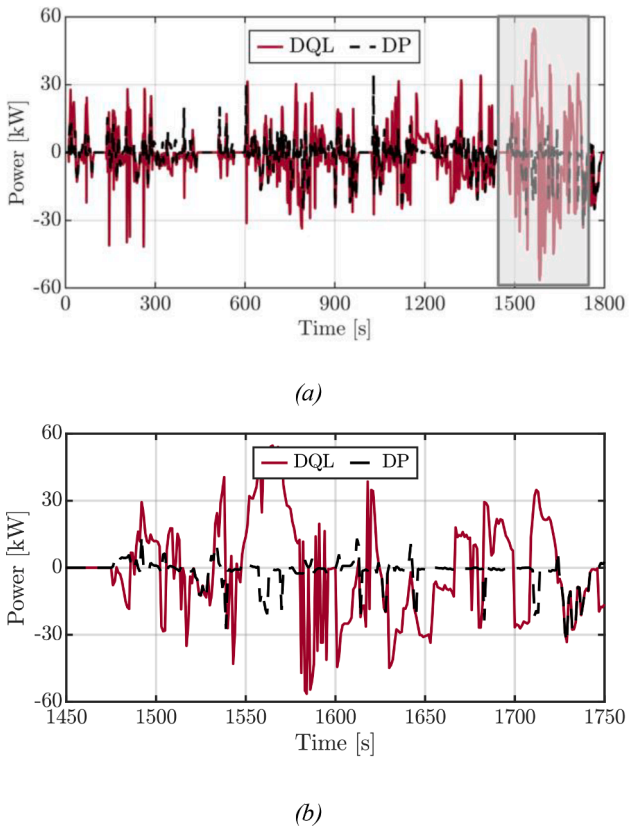


Fig. 6. (a) EM power plotted as a function of time for the DQL agent (red solid line) and DP (black dotted line) on the WLTC. (b) Enlargement of the highlighted area above. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

4. Simulation results

The DQL agent, featuring the rewards presented in Section 3, was trained on the WLTC [44], depicted in Fig. 5, and the results are compared with the global optimum provided by DP. In Section 4.1, the results of the DQL agent featuring the *Case 1* reward will be described more in detail to assess the training effectiveness. Section 4.2, instead, will show the effects of the different rewards on the performance of the DQL agent in selecting the most suitable reward. Finally, in Section 4.3, the performance of the selected agent will be presented on the US06 cycle and on a wide range of well-known type-approval and Real Driving Emission (RDE) compliant driving cycles [45], to assess the agent robustness. The WLTC was chosen for the training of the agent since it is as representative as possible of real-world driving patterns and comprises three different driving scenarios, i.e., urban, rural, and highway. It is noteworthy that testing a PHEV, such as the case study, only in charge-sustaining logic might be limiting. This approach can be regarded as appropriate, though, as the aim of this work is to provide an assessment of how different rewards affect agent performance. Moreover, the fuel economy of a PHEV in charge-sustaining logic is significant for car manufacturers since the UNECE Regulation 83 [44] requires performing a WLTC with minimum battery SoC level while ensuring charge sustainability.

4.1. Training results on WLTC

In this section, the performance of the DQL agent featuring the *Case 1* reward as defined in Eq. (9), is assessed by comparing it with the DP on the WLTC. From Fig. 5, which illustrates the selected operating modes for both control strategies, it can be observed that the logic controlling

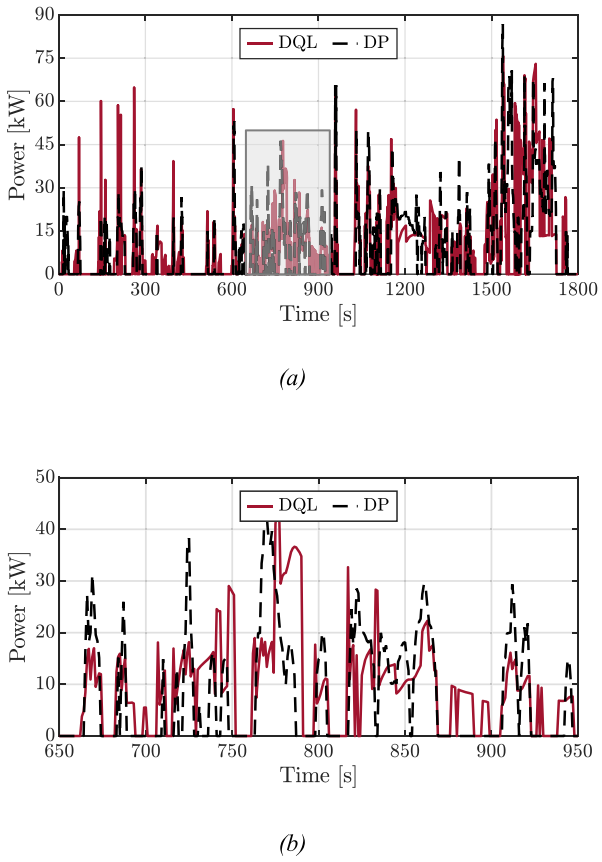


Fig. 7. (a) ICE power plotted as a function of time for the DQL agent (red solid line) and DP (black dotted line) on the WLTC. (b) Enlargement of the highlighted area above. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

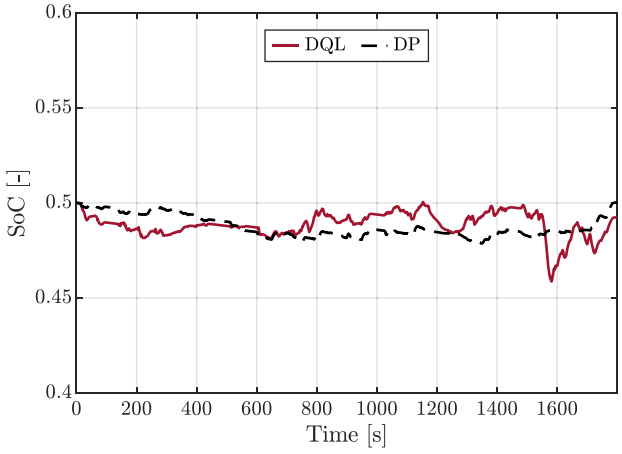


Fig. 8. SoC profile plotted as a function of time for the DQL agent (red solid line) and the DP (black dotted line) on the WTC. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

the engine switching on of the two control strategies shows consistent patterns over the entire driving cycle.

Fig. 6, 7, and 8 provide more granularity by showing the results obtained by the DQL agent (red solid line) and by the DP (black dotted line). Figs. 6(a) and 7(a) display the power provided by the EM and ICE, respectively, during the WLTC, while Figs. 6(b) and 7(b) provide an enlarged view of the highlighted areas. It can be seen that over the WLTC

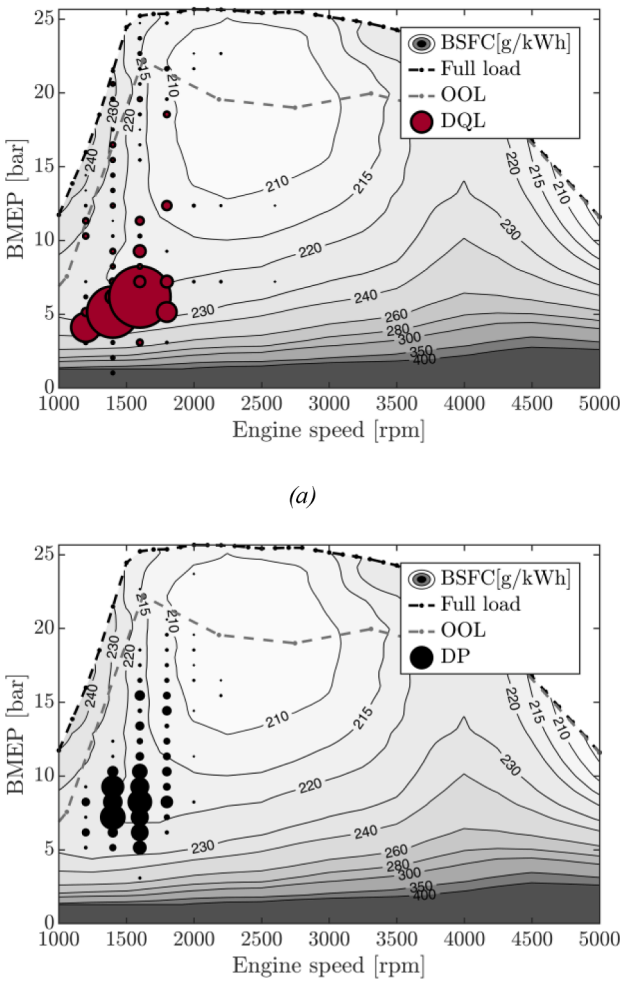


Fig. 9. Comparison of the time distribution of the engine operating points on the WLTC reported on the engine BSFC map - (a) DQL agent (b) DP.

the DQL tends to rely more heavily on the EM for vehicle propulsion. This is particularly evident in the last section of the cycle, as displayed in Fig. 6(b), where the power provided by the EM is constantly higher for

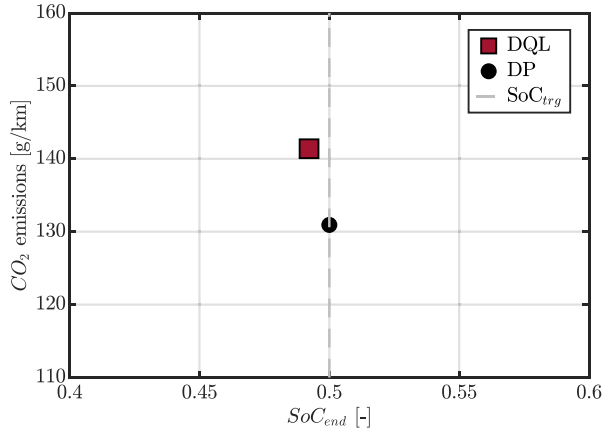


Fig. 10. Comparison between DQL agent (red square) and DP (black circle) on the WLTC: trade-off between CO_2 emissions and final SoC. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 2
Main hyperparameter values.

Discount factor γ	0.99	
Hidden layers	2	
Neurons per each layer	64 (layer 1)	32 (layer 2)
Learning rate	0.001	
Mini-batch size	64	
Experience replay size	18,000	
Target Q-value update frequency	1800	

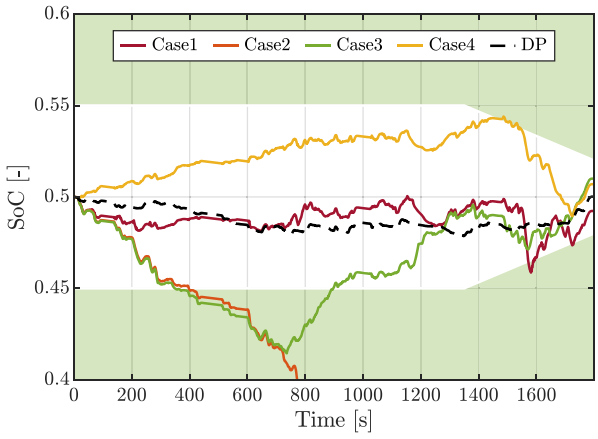


Fig. 11. SoC profiles of the analyzed cases benchmarked against the global optimum provided by the DP on the WLTC. The green area identifies the SoC boundaries defined for the Case 2 reward. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

the DQL, in absolute value, compared to DP. As a consequence, the negative peaks of the EM power are generally higher, in absolute value, for the DQL if compared to the DP as the DQL needs to recharge the battery to ensure charge sustainability.

This behavior can be more clearly seen in Fig. 8 where the SoC profiles of the two strategies are plotted as a function of time. The SoC profile of the DP strategy seems to be more stable while the DQL presents a more rapidly changing behavior. This means not only that the DQL relies more on the EM for propulsion but also employs more often the ICE to recharge the battery. This leads to a different engine operation during the cycle if compared to the DP, as can be observed from Fig. 9 where the time distribution of the engine operating points is reported on the engine Brake Specific Fuel Consumption (BSFC) map: the larger the circle, the higher the time spent by the engine in that operating zone. As already noticed from Fig. 6, it is evident that the DQL exploits the engine mainly at part load, while the DP tends to use the engine at a higher load, guaranteeing a higher efficiency. This discrepancy may lead to the slightly higher fuel consumption of the DQL as illustrated by Fig. 10, where the trade-off between CO_2 emissions and the final SoC value is shown. Although the operation of the DQL may seem diminished if compared to DP (the CO_2 emissions of the DQL are 7.6 % higher, while the battery is slightly depleted at the end of the driving cycle), it should be noted that the DP optimization knows in advance the entire driving mission profile and represents the best achievable results. Therefore, the performance of the trained DQL can be considered sub-optimal in guaranteeing charge sustainability while optimizing fuel economy.

4.2. Sensitivity analysis on the reward

This section analyzes and compares the effects of the rewards introduced in Section 3.3 on the performance of the DQL agent over the WLTC. The four cases are assessed in terms of CO_2 emissions reduction, charge sustainability, and training convergence speed of the algorithm.

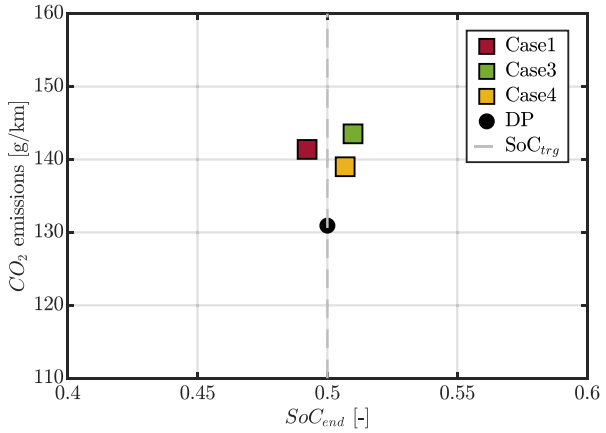


Fig. 12. Comparison between DQL agent featuring three different rewards and DP optimization: trade-off between CO_2 emissions and final SoC on the WLTC.

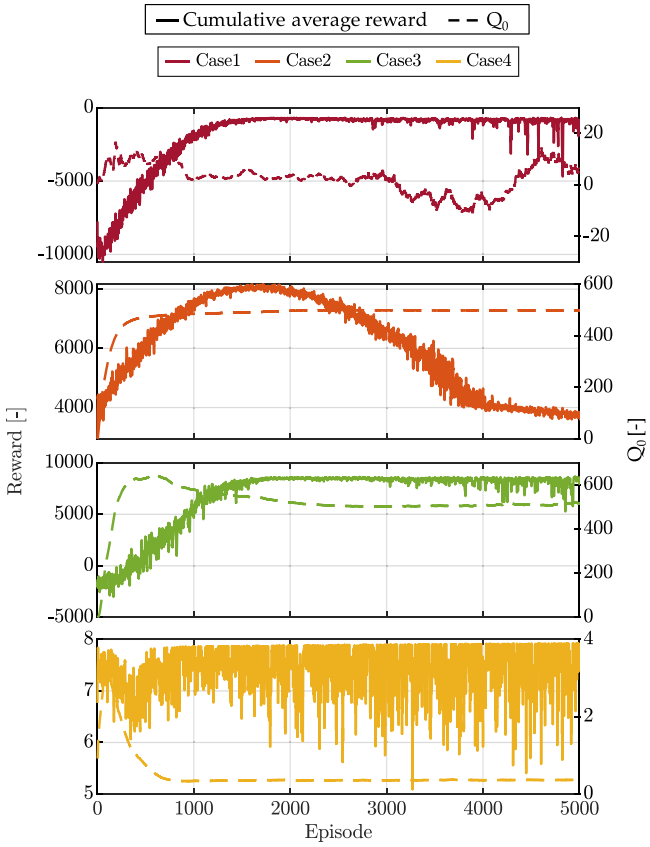


Fig. 13. Training performance of the four tested rewards in terms of average reward (solid line) and initial Q-value (dotted line) on the WLTC.

In an RL framework, the training performance is also affected by the values of the hyperparameters, which can be manually tuned or optimized. In this work, the values of the hyperparameters, as shown in Table 2, were fine-tuned for *Case 1* to reach convergence and subsequently used for the other cases as well. For what concerns the factors k_1, k_2 and k_3, they were individually tuned for each case using a genetic algorithm to maximize the cumulative reward in a charge-sustaining logic.

The SoC profiles are plotted in Fig. 11 for all the considered cases and compared to the optimal profile provided by DP, while Fig. 12 depicts the trade-off between CO_2 emission and the final SoC value. In Fig. 11, the green area delimiting the SoC boundaries is valid only for *Case 3*

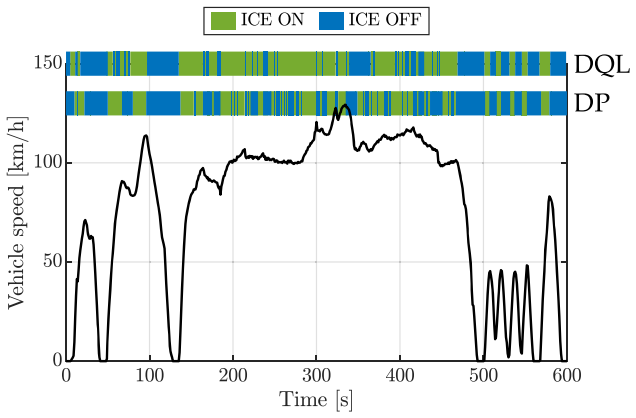


Fig. 14. Operating modes selected by DQL and DP strategies on the US06 cycle.

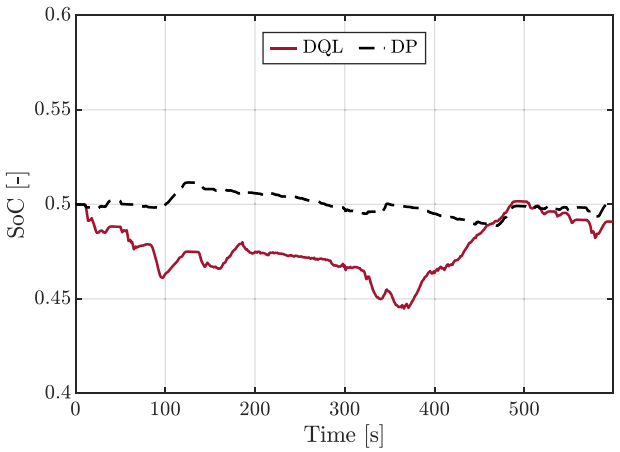


Fig. 15. SoC profile plotted as a function of time for the DQL agent (red solid line) and the DP (black dotted line) on the US06 cycle. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

(green line): if the SoC takes a value outside of the boundaries, a penalty is added to the reward. Apart from *Case 2* (orange line) where the agent selects a policy that excessively discharges the battery, the other cases ensure battery charge sustainability. Thus, it appears that applying an SoC penalty to the reward only at the end of each episode, i.e., a driving cycle, as done in *Case 2*, may not be sufficient to obtain an optimized policy. On the contrary, *Case 1* (red line) presents the most similar behavior to the DP (black dashed line), even if the *Case 4* agent has the best CO_2 emissions/final SoC trade-off (see Fig. 12). However, in *Case 4*, the *s* factor must be a-priori tuned (see Eq. (12)), making this approach sensitive to the mission profile. It should be noted that, in *Case 1*, the SoC profile is confined in a smaller interval around the target if compared to *Case 3* and *Case 4*. This is due to the penalty introduced in the reward formulation which may restrain the agent from choosing an action that leads to a deviation, even if momentary, from the target SoC.

Additional considerations can be done from Fig. 13 which displays the cumulative reward averaged over 5 episodes (solid line) and the initial *Q*-value, namely Q_0, (dotted line) for each analyzed case. The cumulative reward represents the sum of the rewards obtained by the agent in an episode, while the Q_0 represents the estimate for the expected future reward at the beginning of the episode. These two values do not coincide because, in the Q_0 formulation, the discount factor γ is introduced to weigh the influence of future rewards. In general, the training can be considered satisfactory if both the cumulative reward and the Q_0 reach stable and high values as quickly as possible. As evident from Fig. 13, *Case 1* (red line) and *Case 3* (green line) present very stable training, reaching a quasi-optimum configuration, as also confirmed by the results in terms of fuel consumption and charge sustainability reported in Fig. 12. On the other side, although for *Case 4* (yellow line) the reward does not seem to increase during the training phase, it obtains the results closest to the DP in terms of CO_2 emissions/final SoC trade-off. This may suggest that the training of the network has a small influence once the optimal *s* has been chosen for the cycle. Lastly, the reward of *Case 2* (orange line) presents a rising trend until approximately 1500 episodes, followed by a decline to values lower than the previously reached maximum. This training behavior leads to unsatisfactory performance in testing: the agent selects a policy that excessively discharges the battery. Since the simulation does not even reach the end of the cycle (it is stopped because the battery is excessively discharged), *Case 2* cannot be compared with the others and is therefore not included in Fig. 12.

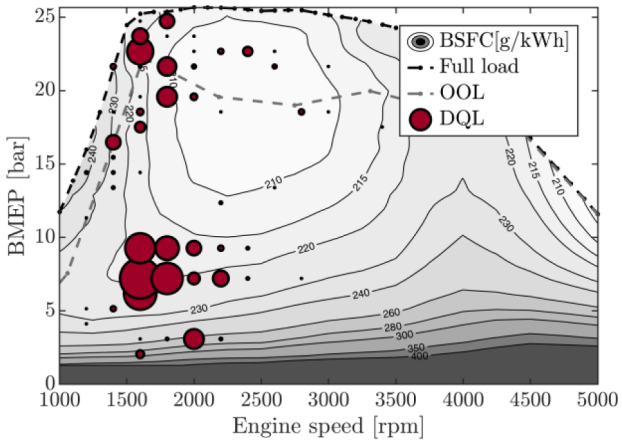
4.3. Agent performance assessment

The sensitivity analysis conducted in Section 4.2 showed that the *Case 4* reward achieves the best CO_2 emissions/final SoC trade-off on the WLTC. However, the ability of this type of reward is hindered by its lack of adaptability to driving cycles different from the one used for the training, since the *s* factor must be separately tuned for each specific driving cycle. Therefore, in this section, the *Case 1* reward is employed to assess the robustness of the DQL agent. The robustness analysis encompasses type-approval and real driving scenarios, offering an evaluation of the agent's performance across a wide range of driving conditions. For the sake of brevity, this section will only showcase the agent performance on the US06 cycle (depicted in Fig. 14) since it features different patterns from the WLTC, while the CO_2 emissions and the final SoC values for each tested cycle are summarized in Table 3.

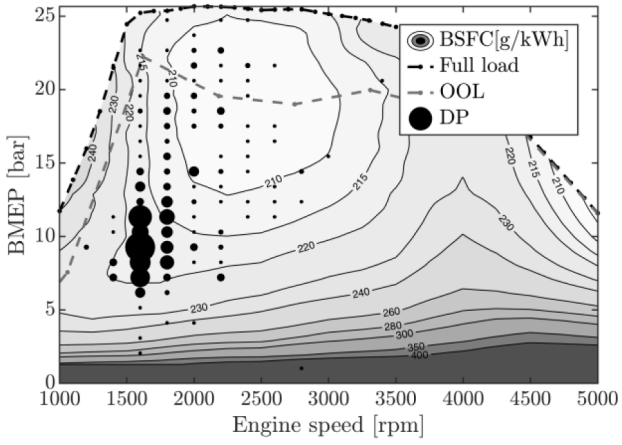
The US06 Supplementary Federal Test Procedure (SFTP) is a driving cycle employed for type-approval purposes in the USA, alongside the standard Federal Test Procedure (FTP) driving cycle. Distinguished by its high-speed segments and aggressive driving patterns, the US06 cycle entails a higher energy demand than the WLTC. Consequently, the US06 cycle was chosen to assess the agent performance in an environment different from the WLTC. Fig. 14 illustrates the selected operating modes for both the DQL agent and the DP, revealing a consistent engine switching logic in both cases, even for this cycle.

Table 3
Robustness assessment of the DQL agent on different driving cycles in terms of CO_2 emission and final SoC compared to the DP. Three main indices, i.e., distance, duration, and average speed, are used to characterize the different driving cycles.

Driving cycle	Distance	Duration	Average Speed	ΔCO_2 emissions compared to DP	Final SoC	ΔFinal battery energy
WLTC (training)	23 km	1800 s	47 km/h	+7.6 %	49.2 %	−108 Wh
Artemis Urban [46]	4.8 km	993 s	17 km/h	+6.0 %	49.5 %	−68 Wh
Artemis Road [46]	17 km	1082 s	57 km/h	+4.7 %	48.6 %	−189 Wh
Artemis Motorway [46]	30 km	1068 s	99 km/h	+3.2 %	47.2 %	−378 Wh
EPA5	44 km	5705 s	45 km/h	+7.2 %	47.9 %	−283 Wh
HFET	17 km	765 s	76 km/h	+10 %	50.6 %	+81 Wh
US06	13 km	600 s	77 km/h	+7.8 %	49.0 %	−135 Wh
RDE cycle	97 km	5926 s	59 km/h	+8.5 %	50.9 %	+122 Wh



(a)



(b)

Fig. 16. Fig. 8: Comparison of the time distribution of the engine operating points on the US06 cycle reported on the engine BSFC map - (a) DQL agent (b) DP.

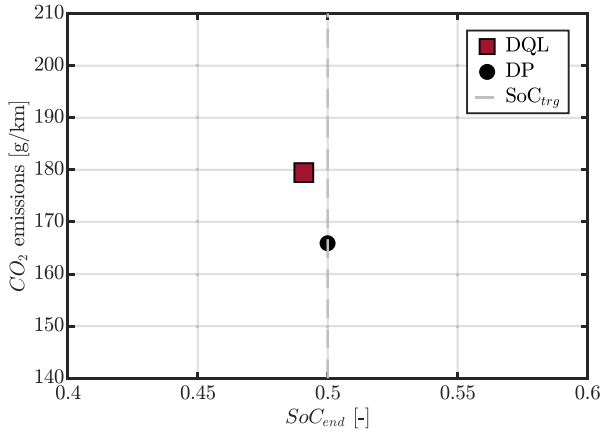


Fig. 17. Comparison between DQL agent (red square) and DP (black circle) on the US06 cycle: trade-off between CO2 emissions and final SoC. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Fig. 15 displays the SoC profiles of the two strategies plotted as a function of time. The main differences between the two strategies can be observed at the beginning and the end of the driving cycle, while the two controllers feature a similar behavior in the intermediate section (from around 100 s to 350 s) resulting in a consistent trend of the SoC profiles. Indeed, at the beginning of the driving cycle, the DQL agent employs the electric motor more frequently, resulting in a greater consumption of electric energy compared to the DP, which leads to a greater SoC discharge. The SoC is recharged only in the final section of the driving cycle to ensure the final charge sustainability.

From Fig. 16, where the time distribution of the engine operating points for both the DQL agent (a) and the DP (b) is depicted, notable differences can be seen between the two strategies in terms of engine utilization patterns. The DQL agent predominantly operates the engine in two zones: at partial load, with the Brake Mean Effective Pressure (BMEP) between 5 and 10 bar, and at full load. On the contrary, the DP algorithm makes the engine work across a wider area on the BSFC map. The discrepancy in the behavior of the two strategies can partially explain the higher CO_2 emissions observed in Fig. 17, which depicts the trade-off between CO_2 emissions and the final SoC. Although the battery is slightly depleted at the end of the driving cycle, the DQL agent leads to an approximately 8 % increase in CO_2 emissions compared to the DP algorithm. Nevertheless, the performance of the DQL agent can be deemed more than satisfactory also on the US06, given that, as previously mentioned, the DP algorithm possesses a priori knowledge of the driving cycle, and its results must be considered optimal.

Table 3 provides an overview of the DQL agent's performance across various driving scenarios, summarizing the differences in CO_2 emissions and final SoC compared to the DP algorithm. This comprehensive analysis highlights the robustness of the agent, as its performance on the driving cycles used for testing closely aligns with the WLTC one. Regarding the charge sustainability, the agent can always satisfy this condition with a SoC deviation of approximately ± 1 % for nearly all the considered driving scenarios. Furthermore, the disparity in CO_2 emissions levels between the DQL agent and the DP ranges from $+3.2$ % (observed in Artemis Motorway with the lowest final SoC) to $+10$ % (observed in HFET with the highest final SoC).

5. Conclusions and further developments

This paper presents a preliminary investigation of the application of Reinforcement Learning (RL) techniques for optimizing the Energy Management System (EMS) of a Plug-in Hybrid Electric Vehicle (PHEV). A Deep *Q*-Learning (DQL) agent was considered, which is value-based, model-free, and off-policy, and the *Q*-value function was modeled employing Deep Neural Networks (DNNs). A sensitivity analysis on different types of reward functions was performed to evaluate their performance in terms of fuel saving, battery charge sustainability, training speed, and stability of the algorithm.

The results of the sensitivity analysis were benchmarked against a global optimal solution provided by Dynamic Programming (DP). The *Case 2* reward, which penalizes the SoC only at the end of the driving cycle, exhibited poor performance in both training and testing, leading to a policy that excessively discharges the battery. The best result was obtained with an ECMS-style reward, which increases the CO_2 emissions by only 6 % if compared to the DP, while slightly charging the battery. However, this approach is case-sensitive due to the introduction in the reward formulation of the *s* factor that must be a-priori tuned. Consequently, the *Case 1* reward was deemed as the best performing one and employed to assess the robustness of the DQL agent in conditions different from the testing ones. The reward was able to achieve performance similar to the training scenario, increasing the fuel consumption by about 7 % on average, compared to the DP, with an average ΔSoC of 1 %.

Future developments of this activity will focus on evaluating the capabilities of this algorithm on a charge-depleting logic along with

testing more complex RL agents, such as Deep Deterministic Policy Gradient (DDPG) and Soft Actor-Critic (SAC). Moreover, the proposed EMS design approach will be applied to a high-fidelity vehicle digital twin relying on a forward dynamic approach.

CRediT authorship contribution statement

Luigi Tresca: Conceptualization, Formal analysis, Methodology, Writing – original draft. **Luca Pulvirenti:** Conceptualization, Methodology, Supervision, Validation, Writing – review & editing. **Luciano Rolando:** Methodology, Supervision, Validation, Writing – review & editing. **Federico Millo:** Supervision, Validation, Writing – review & editing.

Declaration of competing interest

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

Data availability

The data that has been used is confidential.

References

[1] IEA, "Largest end-uses of energy by sector in selected IEA countries, 2018 – Charts – Data & Statistics - IEA." Accessed: May 25, 2022. Available: https://www.iea.org/data-and-statistics/charts/largest-end-uses-of-energy-by-sector-in-selected-iea-countries-2018.

[2] ICCT, "Fit for 55: a review and evaluation of the European Commission proposal for amending the CO2 targets for new cars and vans - International Council on Clean Transportation." Accessed: May 25, 2022 Available: https://theicct.org/publication/fit-for-55-a-review-and-evaluation-of-the-european-commission-proposal-for-amending-the-co2-targets-for-new-cars-and-vans/.

[3] L. Pulvirenti, L. Tresca, L. Rolando, F. Millo, Eco-driving optimization based on variable grid dynamic programming and vehicle connectivity in a real-world scenario, Energies 16 (10) (2023) 4121, https://doi.org/10.3390/EN16104121.

[4] C. Corradi, E. Sica, P. Morone, What drives electric vehicle adoption? Insights from a systematic review on European transport actors and behaviours, Energy Res. Soc. Sci. 95 (2023) 102908, https://doi.org/10.1016/J.ERSS.2022.102908.

[5] A. Sciarretta, L. Guzzella, Control of hybrid electric vehicles, IEEE Control Syst. 27 (2) (2007) 60–70, https://doi.org/10.1109/MCS.2007.338280.

[6] D.D. Tran, M. Vafaeipour, M. El Baghdadi, R. Barrero, J. Van Mierlo, O. Hegazy, Thorough state-of-the-art analysis of electric and hybrid vehicle powertrains: topologies and integrated energy management strategies, Renew. Sustain. Energy Rev. 119 (2020), https://doi.org/10.1016/j.rser.2019.109596. Elsevier Ltd.

[7] A. Biswas, A. Emadi, Energy management systems for electrified powertrains: state-of-the-art review and future trends, IEEE Trans. Veh. Technol. 68 (7) (2019) 6453–6467, https://doi.org/10.1109/TVT.2019.2914457.

[8] D.P. Bertsekas, Dynamic programming and optimal control, Athena Sci. (2005).

[9] G. Paganelli, General supervisory control policy for the energy optimization of charge-sustaining hybrid electric vehicles, JSAE Rev. 22 (4) (2001) 511–518, https://doi.org/10.1016/S0389-4304(01)00138-2.

[10] L.S. Pontryagin, Mathematical Theory of Optimal Processes. Mathematical Theory of Optimal Processes, 2018, https://doi.org/10.1201/9780203749319.

[11] L. Pulvirenti, L. Rolando, F. Millo, Energy management system optimization based on an LSTM deep learning model using vehicle speed prediction, Transp. Eng. 11 (2023), https://doi.org/10.1016/j.treng.2023.100160.

[12] C.E. García, D.M. Prett, M. Morari, Model predictive control: theory and practice—A survey, Automatica 25 (3) (1989) 335–348, https://doi.org/10.1016/0005-1098(89)90002-2.

[13] H. Wang, Y. Huang, A. Khajepour, Q. Song, Model predictive control-based energy management strategy for a series hybrid electric tracked vehicle, Appl. Energy 182 (2016) 105–114, https://doi.org/10.1016/j.apenergy.2016.08.085.

[14] K. Williams, "Real-time stochastic predictive control for hybrid vehicle energy management," ArXiv, 2018.

[15] I.H. Sarker, Machine learning: algorithms, real-world applications and research directions, SN Comput. Sci. 2 (3) (2021) 1–21, https://doi.org/10.1007/S42979-021-00592-X/FIGURES/11.

[16] F. Millo, L. Rolando, L. Tresca, L. Pulvirenti, Development of a neural network-based energy management system for a plug-in hybrid electric vehicle, Transp. Eng. 11 (2023), https://doi.org/10.1016/j.treng.2022.100156.

[17] R.M. Schmidt, "Recurrent neural networks (RNNs): a gentle introduction and overview," 2019, Accessed: Jul. 17, 2023. Available: https://arxiv.org/abs/1912.05911v1.

[18] L. Pack Kaelbling, M.L. Littman, A.W. Moore, and S. Hall, "Reinforcement learning: a survey," 1996.

[19] C.J.C.H. Watkins, P. Dayan, Q-learning, Mach. Learn. 8 (3) (1992) 279–292, https://doi.org/10.1007/BF00992698.

[20] V. Mnih et al., "Playing atari with deep reinforcement learning," 2013, Accessed: Jul. 17, 2023. Available: https://arxiv.org/abs/1312.5602v1.

[21] Y. Lecun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444, https://doi.org/10.1038/nature14539.

[22] T.P. Lillicrap, J.J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, Continuous control with deep reinforcement learning, in: 4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings, 2015. Accessed: Jul. 18, 2023. Available: https://arxiv.org/abs/1509.02971v6.

[23] T. Haarnoja, A. Zhou, P. Abbeel, S. Levine, Soft actor-critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor, in: 35th International Conference on Machine Learning, ICML 2018 5, 2018, pp. 2976–2989. Accessed: Jul. 17, 2023. Available: https://arxiv.org/abs/1801.01290v2.

[24] B. Xu, F. Malmir, D. Rathod, Z. Filipi, Real-Time reinforcement learning optimized energy management for a 48 V mild hybrid electric vehicle. SAE Technical Papers, SAE International, 2019, https://doi.org/10.4271/2019-01-1208.

[25] B. Xu, D. Rathod, D. Zhang, A. Yebi, X. Zhang, X. Li, Z. Filipi, Parametric study on reinforcement learning optimized energy management strategy for a hybrid electric vehicle, Appl. Energy 259 (2020), https://doi.org/10.1016/j.apenergy.2019.114200.

[26] T. Liu, Y. Zou, D. Liu, F. Sun, Reinforcement learning of adaptive energy management with transition probability for a hybrid electric tracked vehicle, IEEE Trans. Ind. Electron. 62 (12) (2015) 7837–7846, https://doi.org/10.1109/TIE.2015.2475419.

[27] T. Liu, X. Hu, S.E. Li, D. Cao, Reinforcement learning optimized look-ahead energy management of a parallel hybrid electric vehicle, IEEE/ASME Trans. Mechatron. 22 (4) (2017) 1497–1507, https://doi.org/10.1109/TMECH.2017.2707338.

[28] A. Musa, P.G. Anselma, G. Belingardi, D.A. Misul, Energy management in hybrid electric vehicles: a Q-learning solution for enhanced drivability and energy efficiency, Energies (Basel) 17 (1) (Dec. 2023) 62, https://doi.org/10.3390/EN17010062.

[29] R. Zou, L. Fan, Y. Dong, S. Zheng, C. Hu, DQL energy management: an online-updated algorithm and its application in fix-line hybrid electric vehicle, Energy 225 (2021), https://doi.org/10.1016/j.energy.2021.120174.

[30] J. Wu, H. He, J. Peng, Y. Li, Z. Li, Continuous reinforcement learning of energy management with deep Q network for a power split hybrid electric bus, Appl. Energy 222 (2018) 799–811, https://doi.org/10.1016/J.APENERGY.2018.03.104.

[31] H. Van Hasselt, A. Guez, D. Silver, Deep reinforcement learning with double Q-learning, in: 30th AAAI Conference on Artificial Intelligence, AAAI 2016, 2015, pp. 2094–2100, https://doi.org/10.1609/aaai.v30i1.10295.

[32] X. Han, H. He, J. Wu, J. Peng, Y. Li, Energy management based on reinforcement learning with double deep Q-learning for a hybrid electric tracked vehicle, Appl. Energy 254 (2019) 113708, https://doi.org/10.1016/J.APENERGY.2019.113708.

[33] R. Lian, J. Peng, Y. Wu, H. Tan, H. Zhang, Rule-interposing deep reinforcement learning based energy management strategy for power-split hybrid electric vehicle, Energy 197 (2020) 117297, https://doi.org/10.1016/J.ENERGY.2020.117297.

[34] T. Li, W. Cui, and N. Cui, "Soft Actor-Critic Algorithm-Based Energy Management Strategy for Plug-In Hybrid Electric Vehicle," *World Electric Vehicle Journal 2022, Vol. 13, Page 193*, vol. 13, no. 10, p. 193, Oct. 2022, doi:10.3390/WEVJ13100193.

[35] W. Huo, T. Zhao, F. Yang, Y. Chen, An improved soft actor-critic based energy management strategy of fuel cell hybrid electric vehicle, J. Energy Storage 72 (2023), https://doi.org/10.1016/j.est.2023.108243, 2352–152.

[36] F. Millo, L. Rolando, L. Pulvirenti, G. Di Pierro, A methodology for the reverse engineering of the energy management strategy of a plug-in hybrid electric vehicle for virtual test rig development, SAE Int. J. Electrif. Veh. 11 (1) (2021), https://doi.org/10.4271/14-11-01-0009.

[37] F. Millo, L. Rolando, M. Andreata, Numerical simulation for vehicle powertrain development, Numer. Anal. - Theory Appl. (2011), https://doi.org/10.5772/24111.

[38] S. Delprat, J. Lauber, T.M. Guerra, J. Rimaux, Control of a parallel hybrid powertrain: optimal control, IEEE Trans. Veh. Technol. 53 (3) (2004) 872–881, https://doi.org/10.1109/TVT.2004.827161.

[39] N. Metropolis, S. Ulam, The Monte Carlo method, J. Am. Stat. Assoc. 44 (247) (1949) 335, https://doi.org/10.2307/2280232.

[40] B. Efron, "Bootstrap methods: another look at the jackknife," vol. 7, no. 1, pp. 1–26, 1979, 10.1214/aos/1176344552.

[41] D.P. Kingma, J.L. Ba, Adam: a method for stochastic optimization, in: 3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings, 2014. Accessed: May 21, 2023. Available, https://arxiv.org/abs/1412.6980v9.

[42] O. Sundström, D. Ambühl, L. Guzzella, On implementation of dynamic programming for optimal control problems with final state constraints, Oil Gas Sci. Technol. 65 (1) (2010) 91–102, https://doi.org/10.2516/ogst/2009020.

[43] O. Sundström, L. Guzzella, A generic dynamic programming Matlab function, in: Proceedings of the IEEE International Conference on Control Applications, 2009, pp. 1625–1630, https://doi.org/10.1109/CCA.2009.5281131.

[44] "Global Technical Regulations (GTRs) | UNECE." Accessed: May 22, 2023. Available: https://unece.org/transport/standards/transport/vehicle-regulations-wp29/global-technical-regulations-gtrs.
[45] European Commission, "Commission Regulation (EU) 2017/1151 of 1 June 2017 of the European Parliament and of the Council on type-approval of motor vehicles with respect to emissions from light passenger and commercial vehicles (Euro 5 and Euro 6)." Accessed: Nov. 17, 2022. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=celex%3A32017R1151.
[46] M. André, The ARTEMIS European driving cycles for measuring car pollutant emissions, Sci. Total Environ. 334–335 (2004) 73–84, https://doi.org/10.1016/J.SCITOTENV.2004.04.070.