POLITECNICO DI TORINO
Repository ISTITUZIONALE

A Cutting-Edge Energy Management System for a Hybrid Electric Vehicle relying on Soft Actor–Critic Deep Reinforcement Learning

Original
A Cutting-Edge Energy Management System for a Hybrid Electric Vehicle relying on Soft Actor–Critic Deep Reinforcement Learning / Tresca, L., Pulvirenti, L., Rolando, L.. - In: TRANSPORTATION ENGINEERING. - ISSN 2666-691X. - ELETTRONICO. - 19:(2025). [10.1016/j.treng.2025.100308]

Availability:
This version is available at: 11583/2998285 since: 2025-03-14T09:46:57Z

Publisher:
Elsevier Ltd

Published
DOI:10.1016/j.treng.2025.100308

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

09 September 2026

Transportation Engineering 19 (2025) 100308

Contents lists available at ScienceDirect

Transportation Engineering

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

Full length article

A Cutting-Edge Energy Management System for a Hybrid Electric Vehicle relying on Soft Actor–Critic Deep Reinforcement Learning

Luigi Tresca[1], Luca Pulvirenti[1], Luciano Rolando *

Politecnico di Torino, Corso Duca degli Abruzzi, 24, Torino, 10129, Italy

ARTICLE INFO

Keywords:
Soft Actor–Critic
Deep Reinforcement Learning
Artificial Intelligence
Energy Management System
Hybrid Electric Vehicle

ABSTRACT

Thanks to its superior learning capabilities and its model-free nature, Reinforcement Learning (RL) is increasingly regarded as an effective solution for addressing complex optimization tasks such as energy management in Hybrid Electric Vehicles (HEVs). In this paper, we implement a Soft Actor–Critic (SAC) agent on a digital twin of a plug-in Hybrid Electric Vehicle (pHEV) operating in charge-sustaining mode. We employ multi-cycle training, which significantly improves the SAC model's ability to generalize across diverse conditions. We fist evaluate the SAC agent capabilities on the Worldwide harmonized Light-duty vehicles Test Cycle (WLTC) by comparing its performance to the global optimum achieved by Dynamic Programming (DP), a local optimization strategy, i.e., Equivalent Consumption Minimization Strategy (ECMS), and a Double Deep Q-Learning (DDQL) algorithm. Furthermore, we test the agent across a broad range of driving cycles to assess its ability to generalize to scenarios beyond those used during training. Simulation results show that the SAC agent achieves results close to the optimal benchmark set by the DP, with CO_2 emissions differing by only 3-4%. The code used in this work is publicly available at: https://github.com/gigitresca/RL-for-EMS-optimization-of-a-Simulink-based-plugin-Hybrid-Electric-Vehicle.git

1. Introduction

As anthropogenic Greenhouse Gas (GHG) emissions continue to accelerate global warming [1,2], the transition to a low- or zero-emission society has become imperative. Mostly due to the adopted policies, overall GHG emissions from member states of the EU decreased by 30% between 1990 and 2021 [3]. However, GHG emissions from the transportation sector are the only ones that have bucked this trend. Except for the substantial drop in 2020 caused by the unprecedented COVID-19 restrictions, GHG emissions from EU transportation experienced a year-over-year rebound of 8.6% in 2021, followed by a further year-over-year growth of 2.7% in 2022 [4].

In this context, exploiting the increasing levels of vehicle connectivity [5,6] along with electrification of the transportation sector not only holds the promise of mitigating GHG emissions but also allows for the improvement of local air quality [7,8]. Battery Electric Vehicles (BEVs) are often advocated as the "silver bullet" [9] since they represent a potential socio-technical solution for reaching transportation sustainability, especially in the light-duty segment. However, significant improvements in battery technology are still required to reach parity with conventional vehicles [10,11]. Hence, inadequate infrastructure combined with limited range and long recharging times of BEVs [12], still makes Hybrid Electric Vehicles (HEVs) and plug-in Hybrid Electric Vehicles (pHEVs) more viable solutions in the mid-term [13].

Nevertheless, to maximize the benefits of powertrain electrification, special attention should be paid to the proper incorporation of the supplementary energy source on board. The vehicle control hierarchy of an HEV must be redesigned by adding a new layer, known as the Energy Management System (EMS). This layer is responsible for deciding the powertrain operating mode and the power distribution between the Internal Combustion Engine (ICE) and the Electric Machines (EMs) [14]. During the past decades, extensive research has been done in EMS to develop strategies that are both implementable in real-time and optimal [15]. Traditionally, energy management strategies have been commonly divided between rule-based and optimization-based methods [16].

Rule-based methods use fixed thresholds derived from experimental knowledge of key parameters to determine the power split among the onboard power sources [17]. Although straightforward to design and implement, these approaches are heavily dependent on expert experience. Moreover, it is not easy to obtain some rules that properly work for a wide range of driving conditions due to the presence of many thresholds and parameters [18].

* Corresponding author.
E-mail address: luciano.rolando@polito.it (L. Rolando).
[1] Co-first Author.

https://doi.org/10.1016/j.treng.2025.100308
Received 26 September 2024; Received in revised form 9 December 2024; Accepted 9 February 2025
Available online 21 February 2025
2666-691X/© 2025 The Authors. Published by Elsevier Ltd. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

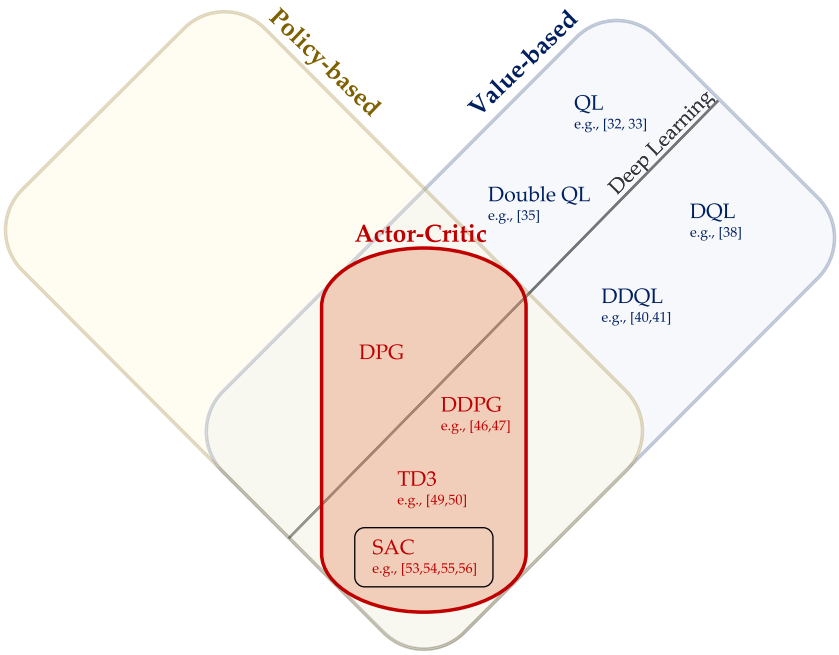

Fig. 1. Venn diagram displaying the three categories of RL agents: policy-based, value-based, and actor–critic. The References highlight their application in the energy management of HEVs.

Optimization-based methods are focused on minimizing an objective function and can be further divided between global and local optimization strategies. Noteworthy global optimization methods include, as examples, Dynamic Programming (DP) [19], and Pontryagin's Minimum Principle (PMP) [20]. They can provide the global optimum but require prior knowledge of the entire vehicle mission profile and involve high computational costs. Given the intrinsic stochastic nature of actual driving cycles, these methods are often employed as theoretical optimal benchmarks [21]. On the other hand, locally optimized methods simplify the global problem by breaking it down into a series of local subproblems, improving the calculation efficiency. They include Equivalent Consumption Minimization Strategy (ECMS) [22], Model Predictive Control (MPC) [23], Evolutionary Algorithms (EA) [24], etc. Although these strategies reduce the computational cost and allow for real-time implementation [25], they may fall into local minima, limiting the maximum exploitation of the optimization techniques.

Beside traditional strategies, data-driven algorithms have recently gained increasing popularity in the energy management of HEVs. In fact, Machine Learning (ML) techniques have permeated into various industrial fields thanks to their incomparable capacity to autonomously find hidden and complex relationships among high-dimensional data. In particular, Reinforcement Learning (RL) algorithms have proven effective in addressing a wide range of challenging optimization and decision-making problems [26,27], such as energy management in HEVs [28]. The great asset of the RL algorithms is their ability to self-learn by directly interacting with an external environment through a trial-and-error process [29]. Hence, RL training entails progressively learning the optimal strategy that minimizes a performance index by rewarding efficient operations and, conversely, penalizing less efficient ones.

Over the last decade, a remarkable amount of research has been conducted into RL-based EMS, resulting in the adoption of a plethora of RL algorithms [30]. These algorithms can be divided into model-based or model-free, depending on whether the agent simulates the transition dynamics of the environment, and on-policy and off-policy, depending on whether the agent adopts the same policy being updated during training. However, as shown in Fig. 1, RL algorithms are generally classified into three categories: policy-based, value-based, and mixed.

The first category of RL algorithms comprises policy search methods, which aim to identify an optimal policy using either gradient-based or gradient-free methods. In these algorithms, a Neural Network (NN), namely the actor, is the policy function that chooses the actions to take given the current state. However, since the policy is an NN with many parameters, searching directly for the optimal policy not only is hampered by the low sample efficiency but also poses challenges that may lead to local minima. As a consequence, these methods are not widely employed in energy management of HEVs.

The second category of algorithms consists of value function methods, which estimate the expected return associated with being in a particular state (referred to as the value). Among value-based algorithms, Q-Learning (QL) [31], which is an off-policy tabular algorithm working in discretized environments, is widely adopted in the literature for the energy management of HEVs. For instance, a tabular Q-learning approach was introduced in [32] accounting for comfort and engine operation requirements in addition to fuel consumption. Or, a Q-learning approach was adopted in [33] for the design of the energy management of a series-parallel HEV that takes into account both fuel consumption and battery life. An advancement in QL methods was achieved when van Hasselt demonstrated that the single estimator employed in QL tends to overestimate the Q-value during the training phase, and thus proposed Double QL which employs a double estimator to manage action selection and value estimation independently [34]. As an example, in [35], two heuristic action execution policies were proposed based on the double QL to improve the vehicle energy efficiency by at least 1%, if compared to standard QL, while maintaining the battery SoC.

Nevertheless, despite the ease of implementation of QL and Double QL agents, their performance is hindered by their tabular-based nature: the dimension of the tables grows exponentially with the number of relevant interacting factors, the so-called "curse of dimensionality" [36]. In this context, the integration of Deep Learning (DL) algorithms into RL significantly contributed to advancements in RL, giving rise to the field of Deep Reinforcement Learning (DRL). Therefore, the performance of the QL method was improved by the introduction of Deep Q-Learning (DQL) [37], which compactly represents high-dimensional observations by employing Deep Neural Networks (DNNs) to approximate the value function. As an example, a DQL algorithm was implemented in the EMS of a hybrid electric bus, as detailed in

[38], which continuously penalized fuel consumption and deviations of the SoC from the target. This approach achieved a 5.6% improvement in fuel economy compared to conventional QL. Although the DQL can improve the capabilities of QL agents and attain satisfactory performance, it is similarly prone to overestimation as the QL method [26]. Hence, the idea behind Double QL was then extended to Double Deep Q-Learning (DDQL) [39], enabling it to handle large-scale function approximation, by employing DNNs for action selection and value estimation. As an example, a DDQL agent was introduced in [40], for an HEV with a dual-motor configuration, demonstrating superior capabilities compared to the DQL approach. Or the Authors previously trained a DDQL agent in [41] for handling the energy management of a pHEV reaching CO_2 emissions only 9% higher than the optimal ones obtained by DP.

Although the adoption of DL successfully solved the "curse of dimensionality" problem associated with value function methods, DQL and DDQL are still hindered by control errors due to the action space discretization. Moreover, in these algorithms, the interval size of the discrete action space must be chosen considering the trade-off between training time and performance. In this framework, the third category of algorithms, which are Actor-Critic (AC) employing both value functions and policy search [42], was proposed to address the aforementioned challenges and achieve continuous control. The policy, namely actor, learns by utilizing feedback provided by the value function, namely critic. Quite recently, the AC approaches have gained increased popularity as they effectively combine the advantages of policy search methods (variance reduction) with the learned value functions (bias introduction) [43]. One of the best-known AC algorithms is Deterministic Policy Gradients (DPG) [44] which involves selecting actions based on a stochastic behavior policy to ensure sufficient exploration while exploiting the current knowledge by simultaneously learning a deterministic target policy.

Analogously to the value function approaches, later work integrated an AC network to the DRL framework, introducing the Deep Deterministic Policy Agent (DDPG) [45]. This approach employs two separate DNNs to approximate the policy and value functions. For instance, a DDPG agent was employed in [46] to address a multi-objective energy management problem, where the agent was continuously penalized for fuel consumption and deviations from the target SoC. Or, in [47], a DDPG agent was integrated into an RB energy management strategy of a heavy-duty HEV that also adopted an organic Rankine cycle for recovering engine waste heat. Nevertheless, DDPG suffers from instability in training, poor convergence, sampling inefficiency, and sensitivity to hyperparameter. Therefore, Twin Delayed Deep Deterministic policy gradient (TD3) [48] was introduced to address these issues. Unlike DDPG, TD3 uses two critic target networks for value estimation and chooses the smaller of the two values. As a consequence, the actor network is updated less frequently than the critic one. As an example, Huang et al. [49] proposed a battery health-aware energy management strategy based on the TD3 method which improved training efficiency by approximately 11% and learning ability by about 24%, compared to a strategy based on DDPG. A TD3-based EMS was proposed in [50] combining a heuristic local controller with a hybrid experience replay method. This approach achieved better fuel economy, faster convergence, and greater robustness compared to conventional value-based and policy-based algorithms.

Nevertheless, among the AC agents, Soft–Actor Critic (SAC) [51] stands out as a prominent method due to its superior learning capability, faster convergence, and better performance compared to the other algorithms [52]. The SAC algorithm is a stochastic and off-policy agent designed to deal with continuous environments and is characterized by entropy maximization which significantly enhances stability and exploration. As an example, Huo et al. [53] proposed an improved SAC algorithm that considered both fuel cell and Li-ion battery life, demonstrating better fuel economy compared to DDPG. Alternatively, a SAC-based EMS was proposed in [54] for a fuel cell HEV, considering the health performance of both fuel cell system and power battery. Or in [55] a SAC agent with automatic entropy tuning was proposed to improve its adaptability to different driving cycles. Finally, the Authors proposed a SAC agent in [56] and performed a sensitivity analysis on different reward functions to evaluate their effectiveness in fuel savings and battery charge sustainability.

According to the surveyed literature, it emerged that, among RL agents, deep model-free methods show significant promise in handling control tasks such as real-time energy management of HEVs. Among them, the information entropy-based SAC algorithm stands out for its remarkable learning capability, convergence speed, and performance. Nevertheless, the majority of works in the literature trained the agent over a single cycle, sometimes also testing the agent's performance on the same driving cycle [50]. Since model-free DRL agents rely on DNNs, they require a substantial amount of stochastic data to prevent overfitting. As a consequence, a DRL agent optimized for a single average driving cycle may perform well during the training phase but not as well under different testing conditions.

1.1. Contributions and objectives

In this paper, we propose a SAC-based energy management strategy for a pHEV operating in charge-sustaining conditions. Differently from other studies available in the literature, we introduce multi-cycle training to enhance the robustness and generalization capability of the SAC agent. The training is performed over an ensemble of 72 driving cycles comprising both type-approval procedures and real driving mission profiles. First, we deeply analyze the training behavior of the SAC agent and compare it to the DDQL used as a benchmark. Then, we evaluate the impact of multi-cycle training by comparing the performance of the SAC agent trained on the multi-cycle ensemble with that of the same agent trained on a single cycle. Finally, we propose a detailed performance assessment of the multi-cycle SAC from an energetic point of view. The SAC algorithm is compared with strategies more widespread in the literature: a global optimization strategy, i.e., DP; a local optimization strategy, i.e., ECMS; and a value-function RL method, i.e., DDQL.

1.2. Paper organization

The rest of the paper is organized as follows: Section 2 briefly introduces the case study, Section 3 formalizes the energy management problem, with special attention to the formulation of DDQL and SAC agents. Then, Section 4 shows the comparison between DDQL and SAC agents during the training process. Section 5 compares the performance of the SAC agent trained over the multi-cycle ensemble to that of the same agent trained over a single cycle. The simulation results are shown on one of the training cycles and one additional test cycle. Then, the SAC agent is compared, in terms of energetic performance, with DP, ECMS, and DDQL. Finally, the paper highlights the main research findings and discusses possible directions for future advancements.

2. Case study

2.1. Vehicle specifications

For this work, we adopted a state-of-the-art pHEV available in the European market as a test case. Fig. 2 schematically sketches, at a high level, the architecture of the powertrain, where an ICE is connected to an EM via an auxiliary clutch. The power actuators are connected to the rear axle via a torque converter and a 9-speed automatic transmission. More granularity is provided by Table 1, where the main characteristics of the vehicle and the powertrain are listed. The engine, whose Brake Specific Fuel Consumption (BSFC) map is depicted in Fig. 3, is a 2-liter diesel engine with 145 kW rated peak power, along with the EM efficiency map.

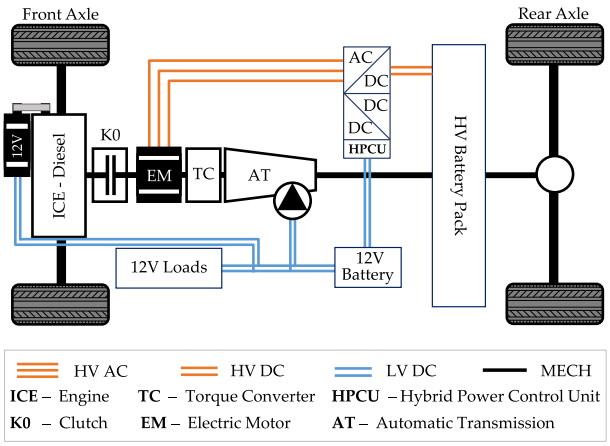

Fig. 2. Powertrain layout: the rear axle is powered by a diesel engine and an EM. MECH denotes the mechanical connections, whereas HV AC, HV DC, and LV DC denote the electrical connections, with high/low voltage alternating/direct current.

Table 1
Main specifications of vehicle and powertrain.

Vehicle	Curb Weight [kg]	2060
	Power [kW] @ 100 km/h	14.9
ICE	Cylinders [–]	4
	Fuel [–]	Diesel
	Displacement [cm^3]	1950
	Max Power [kW] @ 3800 rpm	143
	Max Torque [Nm] @ 1600–2800 rpm	400
	Compression Ratio	15.5:1
	Emissions Standard	Euro 6d-Temp
EM	Type	PM Synchronous Motor
	Max Power [kW] @ 2000 rpm	90
	Max Torque [Nm] @ 1750 rpm	440
	Max Speed [rpm]	6000
Transmission	Type	AT w/Torque Converter
	Gears [–]	9
HV Battery	Type	Li-NMC
	Rated Voltage [V]	365
	Capacity [kWh]/[Ah]	13.5/37

2.2. Virtual test rig development

We built the virtual test rig of the vehicle using data gathered during an extensive experimental campaign conducted on the real vehicle. We built the model without direct access to the vehicle EMS, as detailed in [57]. For this work, which is aimed at assessing the potential of an RL-based EMS, we developed a simplified version of the digital twin in MATLAB®/Simulink®, relying on a backward kinematic approach. Thanks to its limited dynamics, the backward kinematic approach is commonly employed for driving cycle simulations, as it offers computational efficiency and reliable fuel consumption predictions over the entire cycle. Furthermore, this approach is particularly well-suited for RL frameworks, ensuring computational feasibility during the training process.

2.3. Driving cycles dataset

We derived the dataset used in this work from [25]. It consists of 80 traces and covers a wide spectrum of driving patterns, including experimental tests and type-approval driving cycles available from the literature. Fig. 4 illustrates the relationship between all cycles and two key energetic indices. The x-axis represents the square of the average vehicle speed, which is related to the average energy demand. The y-axis displays the product of the average vehicle speed and acceleration, which is related to the aggressiveness of the driving cycle (see [25] for more details of these indices). The gray area is obtained by leveraging the real-world data, available in [58], collected from the field operation of a pHEV driven for 47.000 km during a two-year period. From a visual comparison, it is evident that the mission profiles adopted in this study are in line with the typical real-world operation of a pHEV.

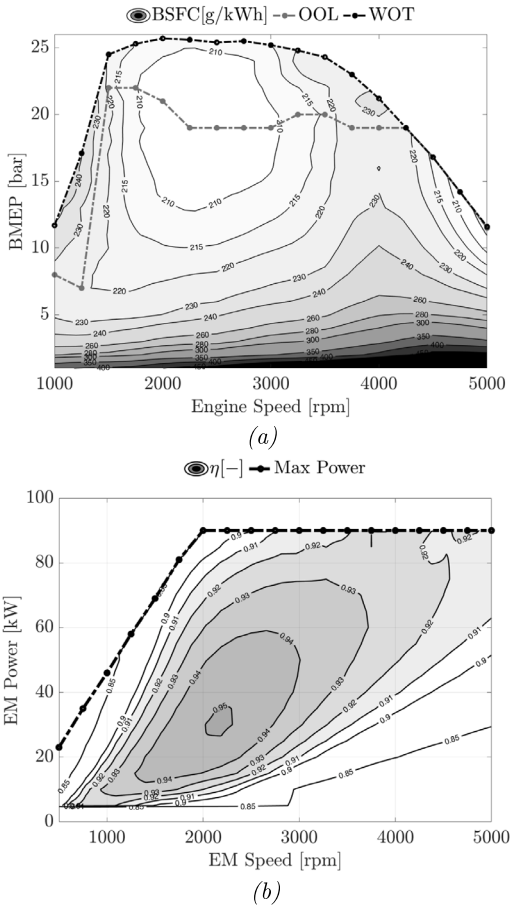

Fig. 3. (a): Brake Specific Fuel Consumption (BSFC) map of the ICE with Wide Open Throttle (WOT) and Optimal Operating Line (OOL) curves; (b): Efficiency map of the EM with max power curve.

We divided the cycles between those used for training the SAC agent (gray) and those used for testing (green). Particular emphasis is given to the two cycles that will be shown in the following Sections. The Worldwide harmonized Light-duty vehicles Test Cycle (WLTC) [59] (red triangle), which is used during training, is not excessively demanding in terms of energy and aggressiveness. On the other hand, the Real-Driving Emissions (RDE)-compliant route [60] (red square), used in the following Sections as a test case, is a more demanding driving cycle.

The RDE driving cycle is illustrated in Fig. 5, depicting vehicle speed over time. The cycle spans roughly 92 minutes and covers a distance of 96 kilometers. We derived the route, depicted in Fig. 5(a), from Portable Emissions Measurement System (PEMS) data and overlaid it on a topographic map. In the test, the urban operation is followed by rural and motorway operations.

3. Energy management

3.1. Problem formulation

The energy management of an HEV can be framed as a constrained and finite-time optimal control problem [19] governed by the following

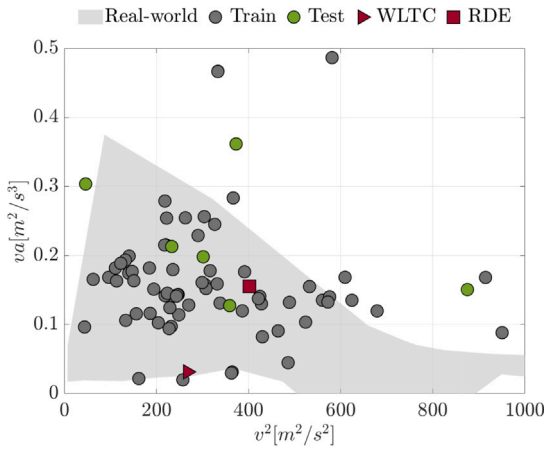

Fig. 4. Driving cycles dataset plotted as a function of squared vehicle speed and the product of vehicle speed and acceleration.

state equation:

$$\dot{x}(t) = f\left(x(t), u(t), t\right) \tag{1}$$

where $x(t) \in \mathbb{R}^n$ and $u(t) \in \mathbb{R}^m$ represent the state and control variables vector, respectively, t denotes the time variable, and f is a function modeling the dynamics of the system. The control policy $u(t) : [t_0, t_f] \in \mathbb{R}^m$ is optimal if it minimizes the cost function J over the time interval $t \in [t_0, t_f]$:

$$J = \phi\left(x(t_f), t_f\right) + \int_{t_0}^{t_f} L\left(x(t), u(t), t\right) dt \tag{2}$$

In this formulation, $L\left(x(t), u(t), t\right) \in \mathbb{R}$ represents the instantaneous cost function, while $\phi\left(x(t_f), t_f\right) \in \mathbb{R}$ defines the terminal cost incurred at the final state. For HEVs, the instantaneous fuel consumption is typically selected as the cost function L, with the optimization problem subject to a set of both local and global constraints related to the physical limitations of the actuators, i.e., ICE and EM, and of the battery:

$$\begin{cases} P_{ice,min}(\omega_{ice}(t)) \le P_{ice}(t) \le P_{ice,max}(\omega_{ice}(t)) \\ P_{em,min}(\omega_{em}(t)) \le P_{em}(t) \le P_{em,max}(\omega_{em}(t)) \\ P_{batt,min}(SoC(t)) \le P_{batt}(t) \le P_{batt,max}(SoC(t)) \end{cases} \tag{3}$$

where ω_{ICE} and ω_{EM} represent the speed of the engine and the electric motor, respectively. Moreover, battery SoC must remain within predefined boundaries $SoC_{min}(t) \le SoC(t) \le SoC_{max}(t)$ and, for an HEV or a pHEV operating in charge-sustaining mode, battery charge sustainability must be ensured: i.e., $SoC(t_f) = SoC(t_i)$.

As already discussed in Section 1, well-known methods, such as DP, ECMS, PMP, etc., have been widely employed to solve this control problem. However, in recent years, RL-based methods have emerged as promising alternatives for this task. The following Sections will be mainly focused on the DDQL and SAC algorithms, alongside a brief overview of the other strategies used for comparison in this study.

3.2. Dynamic programming

Dynamic Programming (DP) [19] is a numerical method for solving multistage decision-making problems. As discussed in Section 1, it belongs to global optimization strategies, and thus provides the optimal solution only within a simulation environment due to its non-causal nature. The DP algorithm is based on Bellman's principle of optimality, and beginning from the final step N, evaluates the optimal cost-to-go function $J_k(x_i)$ at each node in the discretized time state space. This process involves moving backward in time while following the sequence of policies:

$$\pi_k^* = \arg\min_{u_k \in U_k} \left(L_k(x_k, u_k) + J_{k+1}\left(f_k(x_k, u_k)\right)\right) \quad k = N-1, N-2, \ldots, 1, 0 \tag{4}$$

As a global optimization algorithm, DP is not suitable for practical scenarios because it requires the a-priori knowledge of future driving conditions and is hindered by the "curse of dimensionality" due to its high computational demands. Despite these limitations, it is widely used as the optimal benchmark for evaluating other optimization techniques.

3.3. Equivalent consumption minimization strategy

The ECMS is a locally optimized control strategy that relies on the instantaneous optimization of the powertrain energy flows to achieve sub-optimal results [61]. In formulas, the equivalent fuel consumption is obtained by summing the engine fuel consumption $\dot{m}_f(t)$ with a "virtual" fuel consumption $\dot{m}_{el}(t)$ associated with the use of the battery. This combined fuel consumption is minimized instantaneously:

$$\dot{m}_{f,eq}(t) = \dot{m}_f(t) + \frac{s(t) P_{batt}(t)}{Q_{LHV}} \tag{5}$$

where $P_{batt}(t)$ is the instantaneous battery power; Q_{LHV} is the fuel lower heating value; $s(t)$ is the instantaneous equivalence factor that converts battery energy consumption into the virtual fuel consumption, thereby estimating the efficiency of future energy conversion.

Subsequent studies expanded the theoretical bases of the ECMS by demonstrating the correlation between ECMS and PMP [62]. However, in order to obtain sub-optimal results, this approach requires the adoption of the appropriate value of equivalence factor $s(t)$, which depends on both the powertrain topology and the specific mission profile [25]. Therefore, despite its instantaneous formulation, the ECMS implicitly relies on future information. If this information is inaccurate, the results may deviate significantly from optimality. Since its initial introduction, various improvements and modifications have been suggested to enable real-time adaptation of the equivalence factor [61].

3.4. Reinforcement learning

Quite recently, RL has joined the field of energy management of HEV as a promising control strategy to achieve real-time global optimum. The fundamental aspect of RL consists in learning through an interactive trial and error approach: the agent observes the outcomes of its actions and subsequently adjusts its behavior based on the reward it receives, i.e., a scalar value rating the goodness of an action. In RL nomenclature, the state is an ensemble comprising all the sufficient information to describe the environment. For the energy management of HEV, this information may include variables useful for the agent to take the best action, such as vehicle speed, battery SoC, or required power. Since the agent operates in a partially random or unknown environment, RL can be formalized as a Markov Decision Process (MDP) [29]. An MDP is a discrete-time stochastic control process where the state at the next step depends only on the current state and is independent of previous states.

As illustrated in Fig. 6, which depicts the typical agent-environment interaction in an RL algorithm, the agent observes a state S from the environment and interacts with it by selecting an action A, depending on the actual policy π. A causes a change in the environment (new state S') and generates a scalar feedback, namely reward r. The transition information $\{S, A, r, S'\}$ is referred to as an experience, and this process is repeated iteratively until the training is complete. The ultimate goal of the agent is to learn an optimal policy π^*, that maximizes the

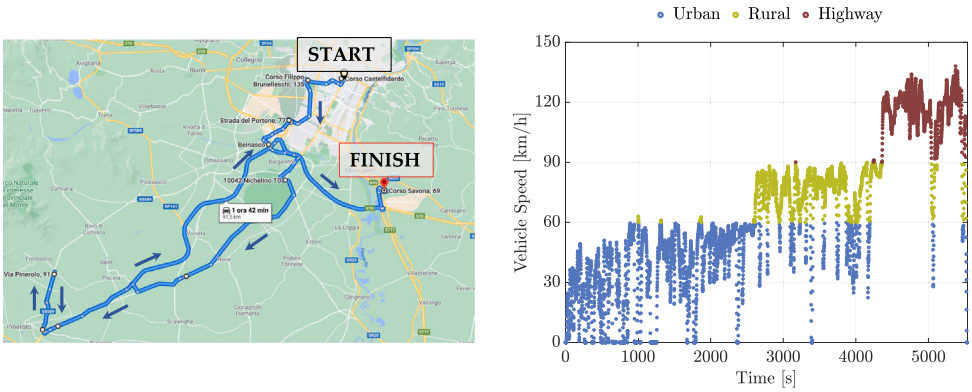

Fig. 5. RDE - (a) Route obtained from PEMS, overlaid on a topographic map (Courtesy of Google Maps); (b) Vehicle speed as a function of time, divided into urban, rural, and highway segments.

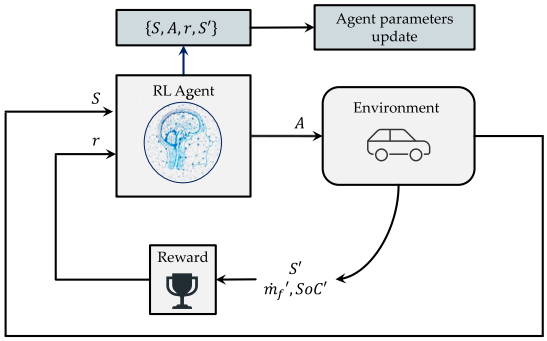

Fig. 6. Schematic representation of the RL operating principles: the agent learns through an interactive trial and error approach.

expected return g_t, expressed as follows:

$$g_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \tag{6}$$

where the discount factor $\gamma \in [0, 1]$ weighs the influence of future rewards.

A key aspect of RL is the fundamental dilemma between exploration and exploitation: trying out random actions to explore the environment or exploiting the current knowledge to maximize the return. In the literature, the ϵ-greedy method is typically used to balance the exploration/exploitation trade-off. It chooses a random action with probability $\epsilon \in [0, 1]$, and a greedy action otherwise. In this method, a suitable ϵ value must be selected to balance the exploration/exploitation trade-off, and, generally, ϵ decreases over time to increase exploitation.

In the next Sections, we will analyze in detail the mathematical formalization of DDQL and SAC agents.

3.4.1. DDQL

Q-Learning (QL) is one of the most widely adopted methods among RL agents, due to its simplicity and effectiveness. It is a value-based and model-free agent able to operate within a discrete action space. QL uses Temporal Difference (TD) learning [63], which combines aspects of Monte Carlo (MC) [64] and DP [19] methods. Similar to MC, TD learning acquires knowledge through direct interaction with the environment. Similar to DP, TD uses a "bootstrapping" approach [65] to estimate the Q-value function.

The Bellman equation is employed to estimate the Q-value function for the actual policy π, which represents the expected cumulative reward (see Eq. 6) obtained by taking action A in state S and then following the policy π. It can be formulated as follows:

$$Q(S, A) = \mathbb{E}_{\substack{A' \sim \pi \\ S' \sim S}} [R(S, A, S') + \gamma Q(S', A')] \tag{7}$$

where R is a scalar function that maps the state transition from S to S' as a result of action A; γ is the discount factor, that determines the present value of future rewards; and $\mathcal{S}$ is the states' distribution, obtained through the agent-environment interaction. It should be noted that the expectation over the next states can be approximated with samples from the agent interaction with the environment, while the expectation over the next actions comes from the current policy π. Since the QL acts in a model-free manner, the expectation operator can be approximated using samples of transitions $\{S, A, r, S'\}$:

$$Q(S, A) \approx r + \gamma Q(S', A') \tag{8}$$

Then, the TD update equation is used to update the Q-value function and is defined as follows:

$$Q(S, A) \leftarrow Q(S, A) + \alpha(r + \gamma \max_{a \in A'} Q(S', a) - Q(S, A)) \tag{9}$$

Since the term $\max_{a \in A'} Q(S', a)$ represents the maximum Q-value at state S', adding r allows to obtain the estimated Q-value of state–action pair (S, A). Thus, the learning rate α acts as a weighting factor between the observed and the estimated Q-value functions.

As discussed in Section 1, the standard QL is hindered by the overestimation of the optimal Q-value and the "curse of dimensionality" as it relies on tables to approximate Q-value functions. Thus, Double Deep Q-Learning (DDQL) was introduced to overcome QL limits. In DDQL, high-dimensional observations can be compactly represented by employing DNNs to approximate the value function, while the overestimation of the optimal Q-value is curbed by training the agent off-policy by adopting a double estimator. The first DNN, known as the behavior Q-value function, interacts with the environment and generates the transition $\{S, A, r, S'\}$. This transition is then stored in a static dataset of transitions $\mathcal{D}$, called experience buffer. The second DNN, known as the target Q-value function is used in the Q-value evaluation and is updated with a custom frequency. This decouples the action selection and value estimation, improving training stability and reducing overestimation of the optimal Q-value. Consequently, the TD target value y can be defined as follows:

$$y = r + \gamma Q_t(S', \operatorname{argmax}_{a \in A'} Q(S', a)) \tag{10}$$

where Q and Q_t represent the behavior and the target Q-value functions, respectively. The update step involves randomly selecting a mini-batch of M transitions $\{S, A, r, S'\}$ from the experience buffer $\mathcal{D}$. A backpropagation algorithm, specifically the Adams optimizer [66], is

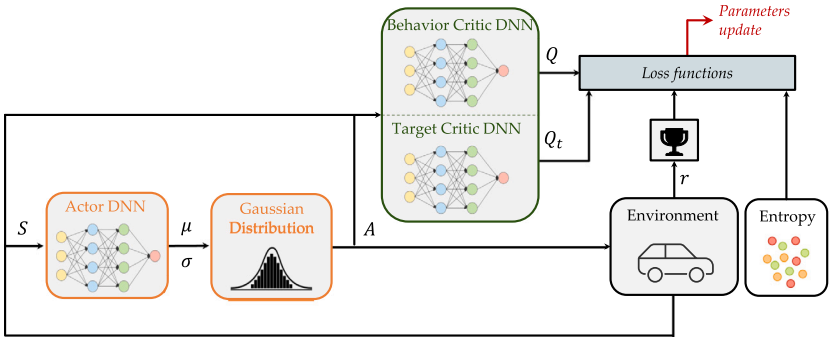

Fig. 7. Schematic representation of the architecture of the SAC agent.

used to train the DNNs. During training, the behavior DNN converges to the target value y. As a result, its parameters are updated by minimizing the loss function:

$$L_Q(\theta) = \mathbb{E}_{\{S,A,r,S'\}\sim D}[(r + \gamma Q_t(S', \underset{a\in A'}{\operatorname{argmax}}\, Q(S', a, \theta), \theta_t) - Q(S, A, \theta))^2] \tag{11}$$

where θ and θ_t are the parameters of the behavior and target DNNs, respectively. At the end of each episode, i.e., a driving cycle, the parameters of the target DNN are updated by copying the optimized parameters from the behavior DNN. Notably, off-policy training enhances sample efficiency by utilizing M different experiences to update the agent's parameters at each time step, unlike on-policy training, which relies on a single experience. Additionally, incorporating an additional network improves training stability, as the parameters of the target DNN are updated solely at the end of each episode.

3.4.2. SAC

As already outlined in Section 1, the DDQL agent can only act in a discrete environment struggling with the challenges posed by large state-action spaces. To overcome these issues, Actor-Critic (AC) agents were introduced, presenting a combination of Q-learning and policy optimization. In this work, we selected the Soft Actor-Critic (SAC) algorithm [51]. SAC is characterized by three fundamental components: an actor-critic framework with separate policy and value function networks, an off-policy formulation that enables efficient use of prior knowledge, and entropy maximization. The latter represents the central feature of SAC as the policy is trained to balance the importance of the expected return and the entropy, which measures the randomness of the policy. This trade-off significantly improves stability and exploration.

The SAC agent is trained off-policy, employing a behavior and a target critic network. The decoupling of the exploration policy from the target policy offers greater flexibility in learning, allowing the agent to collect data with one policy while updating its knowledge with another, similar to the DDQN.

Fig. 7 schematically shows the training loop of the SAC agent. Differently from more widespread deterministic policies, the exploration strategy is handled by a stochastic policy function allowing a proper exploration of the state-action space. Therefore, it provides the mean value and the standard deviation of the action A, which together define a Gaussian distribution, with the mean representing the action that has the highest value.

On the other hand, the critic assesses the quality of the action taken by the actor using the TD error. Both the actor and the critic continuously learn through interaction: the actor adjusts the policy to outperform the critic's expectations, while the critic updates the value function to assess the evolving policy. This iterative loop continues until externally interrupted, making this approach especially well-suited for continuous tasks.

As already noted, the main novelty of the SAC agent relies on the use of the information entropy, which is expressed as follows:

$$H(\pi(\cdot|S)) = -\log \pi(A|S) \tag{12}$$

The larger the policy entropy H, the higher the randomness of the policy, allowing a higher exploration of the state-action space.

Differently from the DDQN, the soft Bellman equation is used. It is obtained by adding an entropy regularization term to the traditional Bellman equation (see Eq.7):

$$Q(S, A) = \mathbb{E}_{\substack{A'\sim\pi \\ S'\sim S}}[R(S, A, S') + \gamma(Q(S', A') + \alpha_T H(\pi(\cdot|S'))] \tag{13}$$

where the parameter α_T is called the entropy regularization coefficient, balancing the impact of the information entropy against the cumulative reward on the agent optimization procedure. As evident from Eq. 13, the agent receives a bonus reward that is proportional to the policy's entropy at that timestep. This feature enhances the robustness of the policy while maintaining its optimality. The α_T parameter is changed dynamically throughout the training, by minimizing the entropy loss function:

$$L_{\alpha_T} = \mathbb{E}_{\{S,A,r,S'\}\sim D}[-\alpha_T \log \pi(A|S, \varphi) - \alpha_T \mathcal{H}] \tag{14}$$

where π is a parametrized policy function with parameters φ and $\mathcal{H}$ is the target entropy, which sets the final desired entropy value.

The entropy regularization coefficient is also adopted in the TD target y, as follows:

$$y = r + \gamma \min_{j=1,2} Q_{t,j}(S', A') - \alpha_T \log \pi(A'|S') \tag{15}$$

In addition to that, the SAC exploits the clipped double-Q trick to mitigate the Q-value overestimation problem, which consists of taking the minimum value of two target Q-value networks differently initialized. The TD target is then used to guide the behavior Q-value networks toward the optimal value, by minimizing the Q-value loss function, i.e., the distance of the two behavior Q-value networks from the TD target:

$$L_{Q_j}(\theta_j) = \mathbb{E}_{\{S,A,r,S'\}\sim D}[(Q_j(S, A, \theta_j) - y(r, S'))^2], \quad j \in \{1, 2\} \tag{16}$$

where Q_1 and Q_2 are the two Q-value networks with parameters θ_1 and θ_2. Finally, the policy network π is updated by maximizing the policy loss function:

$$L_\pi(\varphi) = \mathbb{E}_{\{S,A,r,S'\}\sim D}[\min_{j=1,2} Q_j(S, A, \theta_j) - \alpha \log \pi(A|S, \varphi)] \tag{17}$$

As for the DDQN, the DNNs are trained through the Adams optimizer.

3.4.3. Application

As illustrated in Fig. 8, we selected the following states: vehicle speed (v_{veh}); vehicle acceleration (a_{veh}); gearbox inlet speed (ω_{gb}); battery SoC; the fraction of the traveled distance over the total one

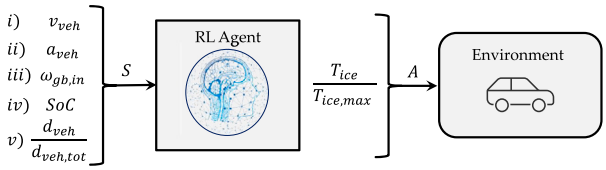

Fig. 8. Application of the RL algorithm to the case study, highlighting the variables selected as states and actions.

(d/d_{tot}). Concerning the action, instead, we chose the engine torque (T_{ice}) divided by the maximum instantaneous available torque $(T_{ice,max})$. The action value always ranges from 0 to 1, thus avoiding unfeasible operating conditions and helping the algorithm converge to a stable solution.

As discussed in Section 3.4, the reward represents the feedback that the agent receives to evaluate the goodness of its actions. Thus, designing an appropriate reward is crucial for guiding the RL agent toward an optimal policy. Unlike DP, RL cannot impose hard constraints on the final SoC value. Therefore, since this work aims to develop an RL-based strategy that guarantees charge-sustaining conditions, the reward function should incorporate soft constraints on the battery's SoC, discouraging excessive deviation from the target value. In this work, we formulated the reward as follows:

$$r = k_1 - k_2 \dot{m}_f \Delta t - k_3 (SoC - SoC_{trg})^2 \tag{18}$$

where $\dot{m}_f$ represents the fuel rate, SoC and SoC_{trg} represent the actual and target SoC, respectively, and k_1, k_2 and k_3 are constants that determine the weight of each term. This reward function aims to minimize ICE fuel consumption while maintaining charge sustainability. It achieves this by applying a continuous soft constraint on the SoC, which is adjusted depending on the deviation from the SoC target. The negative sign converts the miniminization problem into a maximization one. While not necessary, a positive or zero offset k_1 can be added to ensure that the reward stays positive. Achieving a balance between fuel saving and SoC deviation is crucial, requiring a careful tuning of the constants k_1, k_2, and k_3 before training. For the DDQL agent, we adopted the parameters values derived from the optimization performed in [41], and then we manually tuned them for the SAC agent.

3.5. Modeling tools and implementation

We assessed the performance of the SAC agent by comparing it with three different optimization algorithms: a global optimization strategy, i.e., DP; a local optimization strategy, i.e., ECMS; and a value-function RL method, i.e., DDQL. All the algorithms were evaluated on the vehicle virtual test rig relying on a backward kinematic approach. We implemented the DP [19] optimization framework adopting an open-source Matlab code developed at ETH-Zurich [67] which solves discrete-time optimal control problems through Bellman's DP algorithm. Within this framework, we selected battery SoC and ICE power as state and control variables, respectively. For the ECMS, we selected the engine power as the control variable. To obtain an optimal ECMS, the equivalence factor was fine-tuned through a genetic algorithm [68] developed in Matlab. Lastly, we developed the RL framework in Matlab/Simulink, employing the Reinforcement Learning Toolbox and the Deep Learning Toolbox [69]. The former offers a set of reliable implementations of RL agents, while the latter provides functions for designing, implementing, and simulating DNNs.

4. Training assessment

In this work, we trained the SAC agent with a dataset of 72 different driving cycles comprising both type-approval procedures and real-driving mission profiles. This multi-cycle training strategy enables the agent to handle a broader spectrum of driving scenarios, thereby increasing its robustness and generalization capability. In the following Sections, we will analyze the training behavior of the SAC agent by benchmarking it with the DDQL agent, previously developed by us in [41]. Then, we will analyze the SAC agent policy, explaining the decision-making process employed by the agent and how it associates states with corresponding actions.

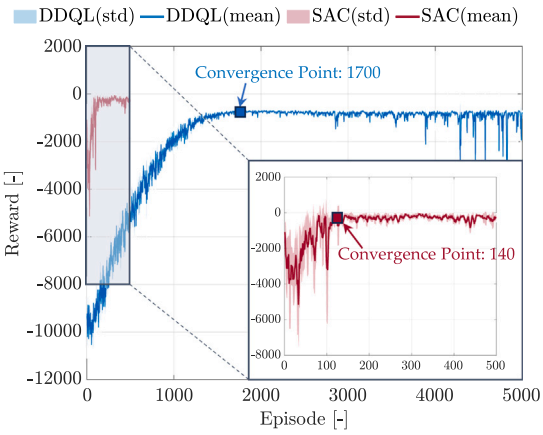

Fig. 9. Training curves as a function of episodes. SAC (red) significantly outperforms DDQL (blue) in terms of convergence speed. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

4.1. SAC vs DDQL

To evaluate the training performance, here we compare the trend of the cumulative reward of SAC and DDQL agents during training. Fig. 9 shows the cumulative reward trend plotted as a function of the training episodes, i.e., driving cycles. For the DDQL (blue) and SAC (red) agents, both the mean value and the standard deviations are plotted. Comparing the two agents, it is striking that SAC outperforms significantly DDQL in terms of convergence speed: it reaches convergence after approximately 140 episodes compared to the 1700 of DDQL. The slower convergence speed of the DDQL may be attributed to the fact that, differently from SAC, the policy is not modeled but extracted from the Q-value network. Moreover, the use of clipped double Q-learning and the addition of noise to the target actions in SAC mitigate the overestimating and overfitting, leading to a significantly improved learning capability. It is worth noting that in the first part (approximately the first 100 episodes) DDQL obtains a smoother and consistent increase in the reward value, while SAC has much wider relative fluctuations if compared to DDQL. This may be attributed to the addition of noise, which promotes exploration, as well as training over the multi-cycle dataset, which introduces additional stochasticity into the training process. The effect, however, is then reduced reaching convergence.

The multi-cycle training method proves particularly effective in training the SAC agent due to its incorporation of policy information entropy regularization. Conversely, the employment of the same training methodology for the DDQL training agent would likely yield less stable training dynamics.

4.2. SAC policy visualization

A key challenge of ML approaches is their black-box nature, which makes them difficult to interpret. In this subsection, we explore how the agent associates states to corresponding actions by using a Gaussian action distribution analysis, in order to provide some insights into the decision-making process employed by the agent. In Fig. 10, the

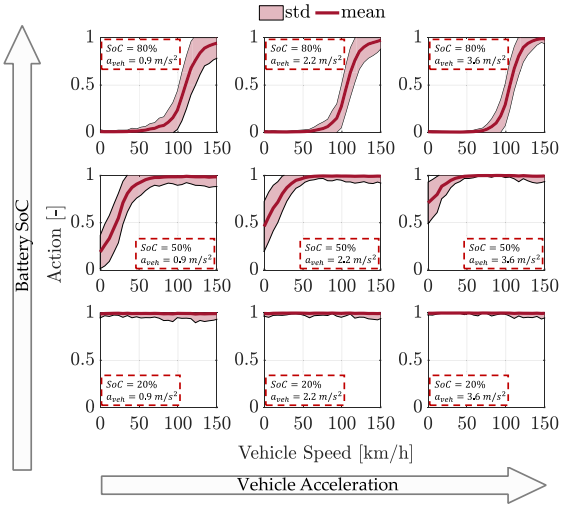

Fig. 10. Action Gaussian distribution at a fixed distance (50%) plotted as a function of vehicle speed, vehicle acceleration, and battery SoC.

Gaussian distribution of the action ($T_{ice}/T_{ice,max}$) is plotted as a function of the vehicle speed at a fixed travel distance (50%). An action equal to 0 means a fully electric mode, while an action equal to 1 means that the engine is operated at full load. The subplots show three different SoC levels, increasing from bottom to top, and three different acceleration levels, increasing from left to right. Focusing on 30% of SoC (bottom subplots), the engine always works at full load to propel the vehicle and recharge the battery. Moving to 50% of SoC (middle subplots), the propulsion strategy mainly depends on the power demand with higher engine exploitation at higher vehicle speeds. At 80% of SoC (upper subplots), for low and moderate vehicle speeds, the agent prioritizes a fully electric mode, and the ICE is used only for higher vehicle speeds. The effect of the vehicle acceleration is more evident at medium SoC levels where the higher the acceleration, the more the agent tends to reach full load at low vehicle speeds.

5. Simulation results

We trained the SAC agent with a dataset comprising both type-approval procedures and real-driving mission profiles. Then, we tested the agent on several mission profiles that were not part of the training dataset. To prove the effectiveness of the multi-cycle SAC, in Section 5.1 we compare its performance, from an energetic perspective, to the scenario where the SAC was trained exclusively on the WLTC. Then, In Section 5.2, we benchmark the multi-cycle SAC against DP, ECMS, and DDQL. For the sake of brevity, only two driving cycles, WLTC and RDE (see Section 2.3), are shown in this Section. Since the WLTC was employed for training the SAC agent, it is regarded as a validation set, while the RDE as a test set.

5.1. SAC single vs SAC multi

In this Section, we compare the performance of the SAC agent trained over several driving cycles (SAC_{multi}) to the case in which the SAC is only trained on WLTC (SAC_{single}). The WLTC is the cycle used by SAC_{single} as it closely represents real-world driving patterns and is among the ensemble of cycles used for training the SAC_{multi}, while the RDE driving cycle is new for both agents. Fig. 11 depicts the SoC trend of the SAC_{single} (green) and SAC_{multi} (red) plotted over the vehicle speed profile for the WLTC (a) and the RDE (b). Although both agents can achieve charge sustainability by the end of the driving cycles, it is notable that the SAC_{single} discharges the battery much more during the cycle compared to the SAC_{multi} whose SoC profile appears more stable.

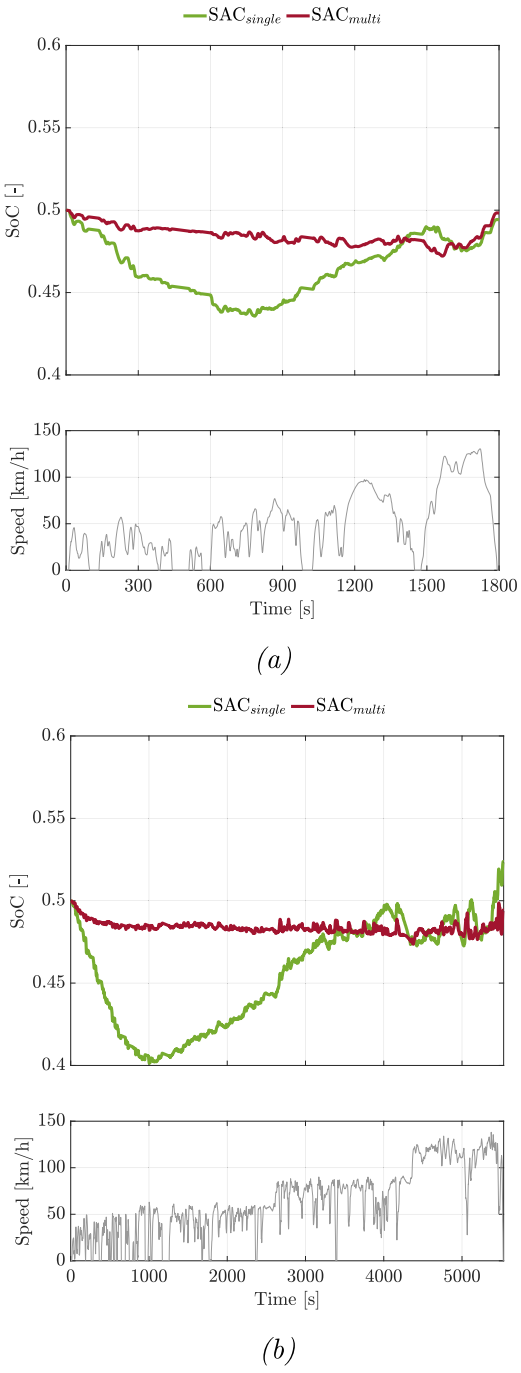

Fig. 11. SoC profiles of SAC trained over only WLTC (green) and over multicycles (red) on the WLTC (a) and RDE (b), along with vehicle speed profiles. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

This suggests that the SAC_{single} relies more on the EM for propulsion but also that it will later require greater use of the ICE to recharge the battery. For the RDE cycle, this behavior is particularly evident as the SAC_{single} shows more SoC fluctuations, especially in the final section of the cycle.

Fig. 12 shows the trade-off between fuel economy and final SoC values, while Table 2 provides more detailed information with the numerical values. It is worth noting that the two agents show comparable performance on the WLTC (triangles), as it is adopted by both agents during the training phase resulting in negligible differences in terms

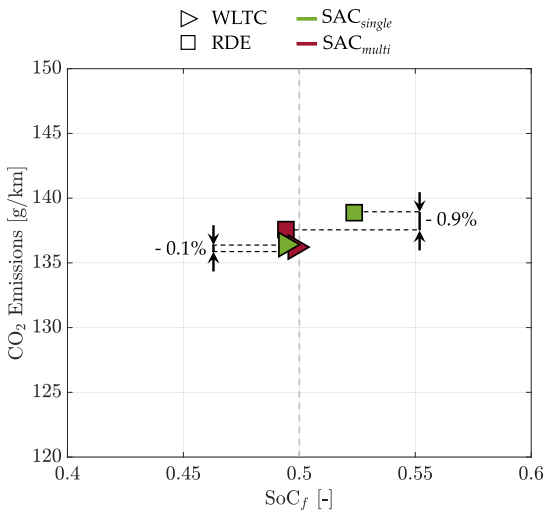

Fig. 12. Comparison between SAC trained over only the WLTC (green), and over multicycles (red) on the WLTC (triangles) and RDE (squares): trade-off between CO_2 emissions and final SoC. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 2
Comparison in terms of final SoC (SoC$_f$) and CO_2 emissions over the WLTC and RDE for the SAC trained over only the WLTC and over multi-cycle. The Δ expresses a percentage difference from the SAC$_{single}$.

	WLTC		RDE	
	CO_2 [g/km]	SoC$_f$ [–]	CO_2 [g/km]	SoC$_f$ [–]
SAC$_{single}$	136.4	0.494	138.9	0.524
SAC$_{multi}$	136.2	0.498	137.6	0.494
Δ	-0.1%	+0.8%	-0.9%	-5.6%

of CO_2 emissions and final SoC values. However, when adopted on a cycle never seen by the agent, i.e., the RDE (squares), SAC$_{multi}$ performs better. It not only allows obtaining better charge sustainability but also reduces CO_2 emissions (−1%). This can be attributed to the fact that the multi-cycle training allows the agent to see more variability during the training phase. Therefore, it can better extrapolate when encountering a cycle different from the ones adopted during the training phase. From now on, we will consider only the SAC$_{multi}$ for further analyses and refer to it simply as SAC.

5.2. SAC vs DDQL vs ECMS vs DP

In this Section, we assess the performance of the SAC agent by comparing it with the DP, ECMS, and DDQL. Fig. 13 depicts the SoC trend of DP (black), ECMS (yellow), DDQL (blue), and SAC (red) plotted as a function of vehicle speed on the WLTC (a) and the RDE (b). Focusing on the WLTC, all strategies can guarantee charge sustainability but the SoC trend is significantly different. It is striking that the SAC obtains a SoC trend that resembles the DP, while the DDQL presents more SoC variations. Finally, the ECMS has a completely different behavior since it charges and then discharges the battery. Focusing on the results of the RDE, DP depletes the battery more during the cycle and then recharges it towards the end. However, the SoC profile of DP remains consistently lower compared to the other strategies. In the RDE, the ECMS has a behavior similar to the one seen in the WLTC: the ECMS is more conservative in using electric propulsion, leading to an increase in the SoC, except in the last section of the cycle in which the battery is depleted.

The different control strategies lead to different fuel consumption as can be seen from Fig. 14. This Figure presents the simulation results for

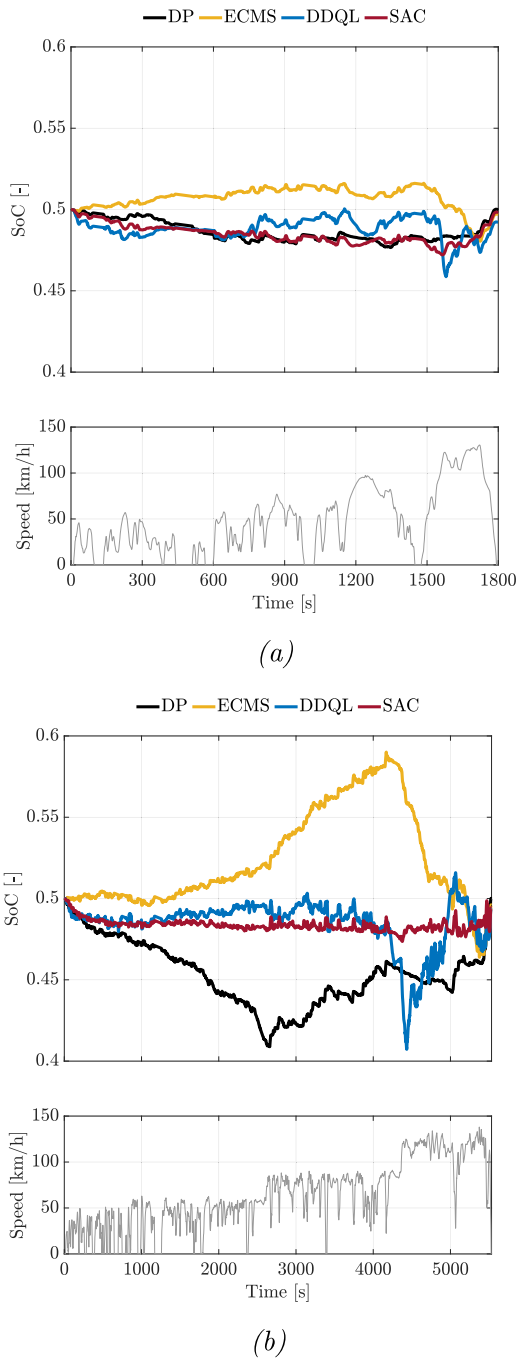

Fig. 13. SoC profiles of DP (black), ECMS (yellow), DDQL (blue), and SAC (red) over the WLTC (a) and RDE (b), along with vehicle speed profiles. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

WLTP (a) and RDE (b) by showing the trade-off between fuel economy and final SoC values for all four strategies. Table 3 provides a more detailed view with the numerical values. By comparing SAC against DP and ECMS, the CO_2 emissions increase by about 4% and 1.2% on average, respectively. This represents a remarkable result considering that DP represents the global optimum, while the ECMS represents a sub-optimal strategy but, for both strategies, the entire driving cycle must be a-priori known. Finally, if compared to the DDQL, the SAC agent allows improving the fuel economy by more than 4%. It should be noted that the advantages of adopting a stochastic agent, such as the SAC, instead of a deterministic one, such as the DDQL, are particularly

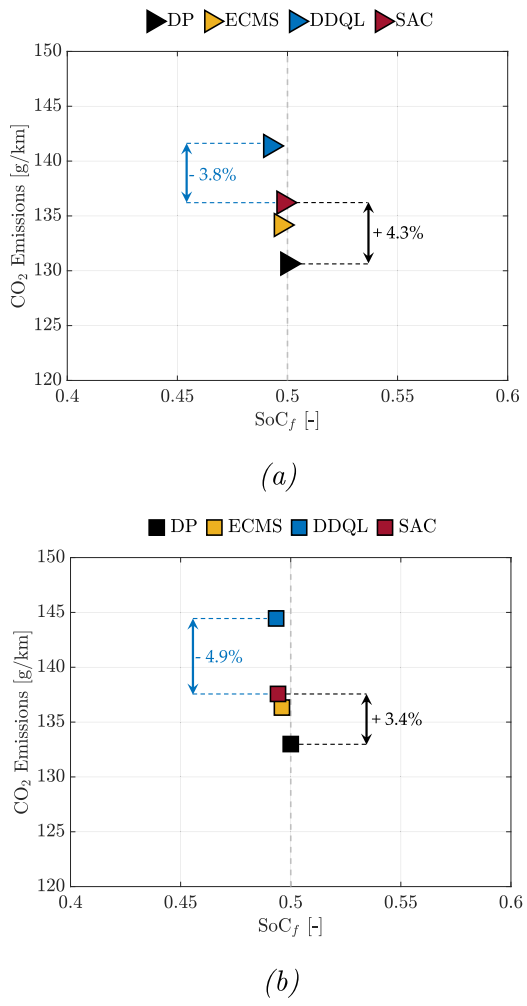

Fig. 14. Comparison between DP (black), ECMS (yellow), DDQL (blue), and SAC (red) on the WLTC (a) and RDE (b): trade-off between CO_2 emissions and final SoC. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 3
Comparison in terms of final SoC (SoC_f) and CO_2 emissions over the WLTC and the RDE for the DP, ECMS, SAC and DDQL. The Δ expresses a percentage difference from the DP.

	WLTC		RDE	
	CO_2 [g/km]	SoC_f [–]	CO_2 [g/km]	SoC_f [–]
DP	130.6	0.500	133.0	0.500
ECMS	134.2	0.497	136.3	0.496
Δ	+2.7%	-0.6%	+2.5%	-0.8%
DDQL	141.4	0.492	144.4	0.493
Δ	+8.2%	-1.6%	+8.6%	-1.3%
SAC	136.2	0.498	137.6	0.494
Δ	+4.3%	-0.4%	+3.4%	-1.1%

evident in the RDE. The action probability distribution of SAC not only enables more effective exploration during training but also improves the algorithm's performance when encountering conditions that were not part of the training data, such as those in the RDE.

We can make further considerations by analyzing the power split chosen by the different control strategies. As an example, Fig. 15 displays the power provided by the ICE during the WLTC for the four analyzed control strategies. As the normalized ICE torque represents the control action of both the RL algorithms, analyzing the ICE actuation can provide useful insights about their behavior. Looking at the interval between 250s and 270s (enlargement above) it can be observed that the SAC behavior is similar to the ECMS and the DP ones. On the contrary, the DDQL has a more nervous behavior reaching higher peak power, maybe related to the discrete action selection. This can be confirmed by looking at the interval between 1450s and 1600s (enlargement below) where it can be noticed that the DDQL continuously switches on and off the ICE. The SAC, instead, has a smoother trend resembling the one obtained by the ECMS and the DP.

A final comparison between the four strategies can be done from Fig. 16, which plots the ICE working points during the RDE on its BSFC map. The engine operating points are indicated by circle markers, where the size of each circle corresponds to the time the engine spends in that specific region. The bigger the circle on the map the more time the engine spent in that operating condition. Comparing the four strategies, it is evident that for the DP and SAC strategies, the ICE is operated in a wider area including regions with lower BSFC. On the contrary, DDQL and ECMS tend to spend more time in specific regions (bigger circles). The behavior of the DDQL may be attributed to its discretized nature, while SAC works with a continuous action space. Finally, it is worth noticing that the SAC mirrors the control logic of the DP approach, but without requiring prior knowledge of the entire vehicle mission profile.

For the sake of brevity, we showed the methodology validation only on two significant driving cycles. However, to demonstrate the robustness of the SAC-based approach in generalizing across varying driving conditions, we extended the analysis to a substantially broader set of driving cycles. The trade-off between SoC and CO_2 emissions for all the tested conditions is detailed in the Supplemental Material Section (see Figure S1 and Table S1).

6. Conclusions and further developments

In an era in which big data and Artificial Intelligence (AI) are already revolutionizing our society, AI algorithms give us the opportunity to deeply reshape the energy management of Hybrid Electric Vehicles (HEVs).

In this article, we proposed an AI-based control strategy for the powertrain of a plug-in Hybrid Electric Vehicle (pHEV). For this application, we chose a Soft Actor-Critic (SAC) agent due to its high convergence rate and robustness. Moreover, we introduced multi-cycle training of the SAC agent over an ensemble of 72 different mission profiles comprising both type-approval procedures and real-driving mission profiles.

The results showed that multi-cycle training significantly enhanced the SAC model's ability to generalize across diverse conditions. We compared the results of the proposed methodology to various reference sets used as a benchmark: a global optimization strategy, i.e., the Dynamic Programming (DP), a local optimization strategy, i.e., the Equivalent Consumption Minimization Strategy (ECMS), and a value-based Deep Reinforcement Learning (DRL) agent, i.e., the Deep Double Q-learning (DDQL).

The SAC agent achieved a relatively small increase in CO_2 emissions compared to the global optimum provided by DP, with a difference of 3-4%. Moreover, the SAC remarkably improved the performance, if compared to the critic-based DDQL (by more than 4%), and achieved comparable results to the ECMS, but without the prior knowledge of the entire vehicle speed profile.

The simulation results highlighted the potential of a SAC-based control strategy to generalize across different mission profiles. Such strategies appear particularly promising for real-world applications, where vehicles encounter varying driving patterns, and conventional optimization methods often fall short due to insufficient knowledge or limited optimization time.

In future developments, we will focus on evaluating the algorithm capabilities within a charge-depleting framework. Moreover, we will investigate the influence of vehicle connectivity data on this control strategy. Eventually, the proposed RL algorithm will be implemented in a virtual test rig modeled with a forward dynamic approach.

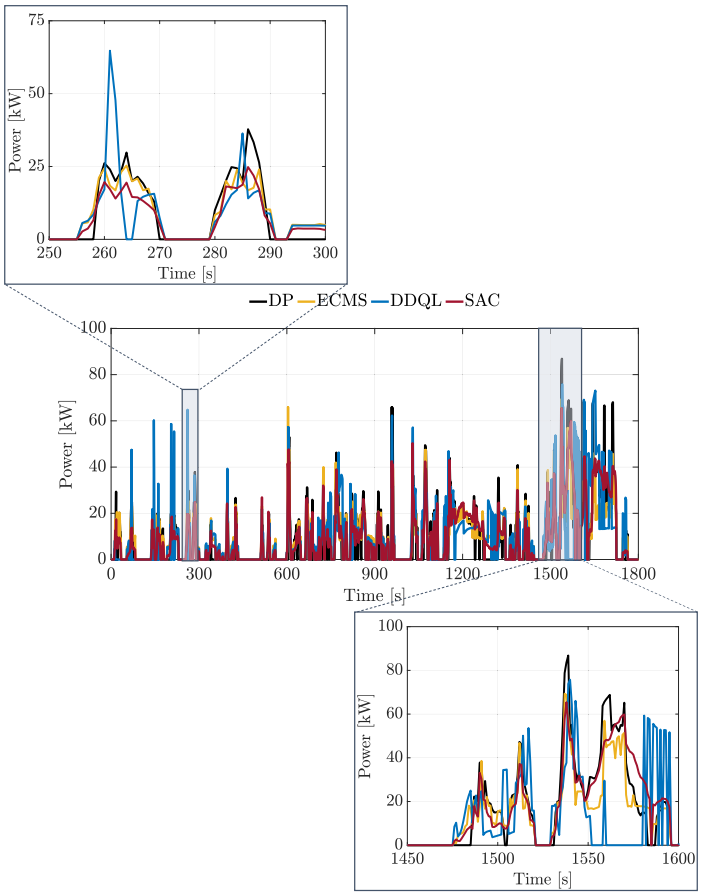

Fig. 15. ICE power plotted as a function of time for the DP (black), ECMS (yellow), DDQL (blue), and SAC (red) on the WLTC, with two enlargements of the highlighted areas. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

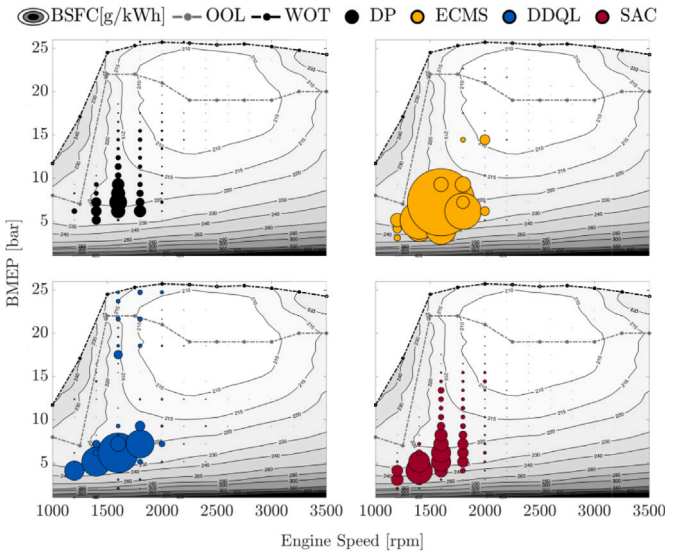

Fig. 16. ICE working points during the RDE on its BSFC map: DP (black), ECMS (yellow), DDQL (blue), and SAC (red). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Abbreviations

The following abbreviations are used in this manuscript:

AC	Actor-Critic
AI	Artificial Intelligence
BEV	Battery Electric Vehicle
BSFC	Brake Specific Fuel Consumption
DDPG	Deep Deterministic Policy Agent
DDQL	Double Deep Q-Learning
DQL	Deep Q-Learning
DNN	Deep Neural Network
DP	Dynamic Programming
DPG	Deterministic Policy Gradients
DRL	Deep Reinforcement Learning
EA	Evolutionary Algorithms
ECMS	Equivalent Consumption Minimization Strategy
EM	Electric Machine
EMS	Energy Management System
GHG	Greenhouse Gas

HEV	Hybrid Electric Vehicle
ICE	Internal Combustion Engine
MC	Monte Carlo
ML	Machine Learning
MPC	Model Predictive Control
NN	Neural Network
OOL	Optimal Operating Line
PEMS	Portable Emissions Measurement System
pHEV	plug-in Hybrid Electric Vehicle
PID	Proportional-Integral-Derivative
PM	Permanent Magnet
QL	Q-Learning
RDE	Real-Driving Emissions
RB	Rule-Based
RL	Reinforcement Learning
SAC	Soft Actor-Critic
SoC	State of Charge
TD	Temporal Difference
TD3	Twin Delayed Deep Deterministic policy gradient
WLTC	Worldwide harmonized Light-duty vehicles Test Cycle
WOT	Wide Open Throttle

Symbols

The following symbols are used in this manuscript:

A	action
CO_2	carbon dioxide
$\mathcal{D}$	experience buffer
g_t	expected return
H	policy entropy
$\mathcal{H}$	target entropy
J	cost function
k_i	constant
$L(\theta_i)$	loss function
$L\left(x(t), u(t), t\right)$	instantaneous cost function
$\dot{m}_{el}$	virtual fuel consumption
$\dot{m}_{eq}$	equivalent fuel consumption
$\dot{m}_f$	engine fuel consumption
P_{batt}	battery power
P_{em}	EM power
P_{ice}	ICE power
Q_{LHV}	fuel lower heating value
$Q(S, A)$	Q-value function
r	reward
R	scalar function
S	state
$\mathcal{S}$	states distribution
$s(t)$	equivalence factor
$u(t)$	control variable
$x(t)$	state variable
α	learning rate
α_T	entropy regularization coefficient
γ	discount factor
ϵ	probability
θ	parameters of DNN
π	policy
$\phi\left(x(t_f), t_f\right)$	terminal cost
$\varphi\left(x(t_f), t_f\right)$	parameters
ω_{em}	EM speed
ω_{ice}	ICE speed

CRediT authorship contribution statement

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

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.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.treng.2025.100308.

Data availability

The authors will like to share the developed code through a GitHub Link.

References

[1] K.O. Yoro, M.O. Daramola, Chapter 1 - CO2 emission sources, greenhouse gases, and the global warming effect, in: M.R. Rahimpour, M. Farsi, M.A. Makarem (Eds.), Advances in Carbon Capture, Woodhead Publishing, 2020, pp. 3–28, http://dx.doi.org/10.1016/B978-0-12-819657-1.00001-3.

[2] U. UNEP, Emissions gap report 2020, UN, 2021, URL https://www.unep.org/emissions-gap-report-2020.

[3] EEA, Total net greenhouse gas emission trends and projections in europe, 2024, online documentation, https://www.eea.europa.eu/en/analysis/indicators/total-greenhouse-gas-emission-trends#:~:text=Net%20greenhouse%20gas%20(GHG)%20emissions,%2Dyear%20reduction%20of%201.9%25.

[4] EEA, Greenhouse gas emissions from transport in Europe, 2024, online documentation, https://www.eea.europa.eu/en/analysis/indicators/greenhouse-gas-emissions-from-transport.

[5] Q. Guo, O. Angah, Z. Liu, X.J. Ban, Hybrid deep reinforcement learning based eco-driving for low-level connected and automated vehicles along signalized corridors, Transp. Res. Part C: Emerg. Technol. 124 (2021) 102980, http://dx.doi.org/10.1016/j.trc.2021.102980.

[6] 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, http://dx.doi.org/10.3390/en16104121.

[7] M. Muratori, Impact of uncoordinated plug-in electric vehicle charging on residential power demand, Nat. Energy 3 (3) (2018) 193–201, http://dx.doi.org/10.1038/s41560-017-0074-z.

[8] E. Kriegler, J.P. Weyant, G.J. Blanford, V. Krey, L. Clarke, J. Edmonds, A. Fawcett, G. Luderer, K. Riahi, R. Richels, et al., The role of technology for achieving climate policy objectives: overview of the EMF 27 study on global technology and climate policy strategies, Clim. Change 123 (2014) 353–367, http://dx.doi.org/10.1007/s10584-013-0953-7.

[9] A. Milovanoff, I.D. Posen, H.L. MacLean, Electrification of light-duty vehicle fleet alone will not meet mitigation targets, Nat. Clim. Chang. 10 (12) (2020) 1102–1107, http://dx.doi.org/10.1038/s41558-020-00921-7.

[10] J. Deng, C. Bae, A. Denlinger, T. Miller, Electric vehicles batteries: requirements and challenges, Joule 4 (3) (2020) 511–515, http://dx.doi.org/10.1016/j.joule.2020.01.013.

[11] I.-Y.L. Hsieh, M.S. Pan, Y.-M. Chiang, W.H. Green, Learning only buys you so much: Practical limits on battery price reduction, Appl. Energy 239 (2019) 218–224, http://dx.doi.org/10.1016/j.apenergy.2019.01.138.

[12] 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, http://dx.doi.org/10.1016/j.erss.2022.102908.

[13] A. Garcia, J. Monsalve Serrano, R.L. Sari, S. Tripathi, Life cycle CO2 footprint reduction comparison of hybrid and electric buses for bus transit networks, Appl. Energy 308 (2022) 118354, http://dx.doi.org/10.1016/j.apenergy.2021.118354.

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

[15] 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, http://dx.doi.org/10.1109/TVT.2019.2914457.

[16] 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) 109596, http://dx.doi.org/10.1016/j.rser.2019.109596.

[17] T. Hofman, M. Steinbuch, R. Van Druten, A. Serrarens, Rule-based energy management strategies for hybrid vehicles, Int. J. Electr. Hybrid Veh. 1 (1) (2007) 71–94, http://dx.doi.org/10.1504/IJEHV.2007.014448.

[18] G. Du, Y. Zou, X. Zhang, L. Guo, N. Guo, Heuristic energy management strategy of hybrid electric vehicle based on deep reinforcement learning with accelerated gradient optimization, IEEE Trans. Transp. Electrification 7 (4) (2021) 2194–2208, http://dx.doi.org/10.1109/TTE.2021.3088853.

[19] R. Bellman, Dynamic Programming, Dover Publications, 1957, URL https://www.bibsonomy.org/bibtex/29cdd821222218ded252c8ba5cd712666/pcbouman.

[20] H. Kaufman, The mathematical theory of optimal processes, by l. s. Pontryagin, v. g. Boltyanskii, r. V. Gamkrelidze, and e. f. Mishchenko. Authorized translation from the Russian. Translator: K. n. Trirogoff, editor: L.. w. neustadt. Interscience publishers (division of john wiley and sons, inc., new york) 1962. viii 360 pages., Canad. Math. Bull. 7 (3) (1964) 500, http://dx.doi.org/10.1017/S0008439500032112.

[21] 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) 100156, http://dx.doi.org/10.1016/j.treng.2022.100156.

[22] G. Paganelli, T.-M. Guerra, S. Delprat, J.-J. Santin, M. Delhom, E. Combes, Simulation and assessment of power control strategies for a parallel hybrid car, Int. J. Automob. Eng. 214 (2000) 705–717, http://dx.doi.org/10.1243/0954407001527583.

[23] B. Kouvaritakis, M. Cannon, Model predictive control: Classical, robust and stochastic, Springer, 2016, http://dx.doi.org/10.1007/978-3-319-24853-0,

[24] E. Zitzler, Evolutionary algorithms for multiobjective optimization: Methods and applications, vol. 63, Shaker Ithaca, 1999, URL https://eckartzitzler.ch/img/publications/zitz1999a.pdf.

[25] 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) 100160, http://dx.doi.org/10.1016/j.treng.2023.100160.

[26] K. Arulkumaran, M.P. Deisenroth, M. Brundage, A.A. Bharath, Deep reinforcement learning: A brief survey, IEEE Signal Process. Mag. 34 (6) (2017) 26–38, http://dx.doi.org/10.1109/MSP.2017.2743240.

[27] G. Pinto, D. Deltetto, A. Capozzoli, Data-driven district energy management with surrogate models and deep reinforcement learning, Appl. Energy 304 (2021) 117642, http://dx.doi.org/10.1016/j.apenergy.2021.117642.

[28] C. Song, K. Kim, D. Sung, K. Kim, H. Yang, H. Lee, G.Y. Cho, S.W. Cha, A review of optimal energy management strategies using machine learning techniques for hybrid electric vehicles, Int. J. Automot. Technol. 22 (2021) 1437–1452, http://dx.doi.org/10.1007/s12239-021-0125-0.

[29] R.S. Sutton, A.G. Barto, Reinforcement learning: An introduction, MIT Press, 2018, URL https://web.stanford.edu/class/psych209/Readings/SuttonBartoIPRLBook2ndEd.pdf.

[30] D. Xu, C. Zheng, Y. Cui, S. Fu, N. Kim, S.W. Cha, Recent progress in learning algorithms applied in energy management of hybrid vehicles: A comprehensive review, Int. J. Precis. Eng. Manufacturing- Green Technol. 10 (1) (2023) 245–267, http://dx.doi.org/10.1007/s40684-022-00476-2.

[31] C.J. Watkins, P. Dayan, Q-learning, Mach. Learn. 8 (1992) 279–292, http://dx.doi.org/10.1007/BF00992698.

[32] 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 17 (1) (2023) 62, http://dx.doi.org/10.3390/en17010062.

[33] S. Ahmadian, M. Tahmasbi, R. Abedi, Q-learning based control for energy management of series-parallel hybrid vehicles with balanced fuel consumption and battery life, Energy AI 11 (2023) 100217, http://dx.doi.org/10.1016/j.egyai.2022.100217.

[34] H. Hasselt, Double Q-learning, Adv. Neural Inf. Process. Syst. 23 (2010) URL https://proceedings.neurips.cc/paper_files/paper/2010/file/091d584fced301b442654dd8c23b3fc9-Paper.pdf.

[35] B. Shuai, Q. Zhou, J. Li, Y. He, Z. Li, H. Williams, H. Xu, S. Shuai, Heuristic action execution for energy efficient charge-sustaining control of connected hybrid vehicles with model-free double Q-learning, Appl. Energy 267 (2020) 114900, http://dx.doi.org/10.1016/j.apenergy.2020.114900.

[36] Y. Bengio, A. Courville, P. Vincent, Representation learning: A review and new perspectives, IEEE Trans. Pattern Anal. Mach. Intell. 35 (8) (2013) 1798–1828, http://dx.doi.org/10.1109/TPAMI.2013.50.

[37] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M. Riedmiller, Playing atari with deep reinforcement learning, 2013, http://dx.doi.org/10.48550/arXiv.1312.5602, arXiv preprint arXiv:1312.5602.

[38] 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, http://dx.doi.org/10.1016/j.apenergy.2018.03.104.

[39] H. Van Hasselt, A. Guez, D. Silver, Deep reinforcement learning with double q-learning, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 30, No. 1, 2016, http://dx.doi.org/10.1609/aaai.v30i1.10295.

[40] 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, http://dx.doi.org/10.1016/j.apenergy.2019.113708.

[41] L. Tresca, L. Pulvirenti, L. Rolando, F. Millo, Development of a deep Q-learning energy management system for a hybrid electric vehicle, Transp. Eng. 16 (2024) 100241, http://dx.doi.org/10.1016/j.treng.2024.100241.

[42] V. Konda, J. Tsitsiklis, Actor-critic algorithms, Adv. Neural Inf. Process. Syst. 12 (1999) http://dx.doi.org/10.1137/S0363012901385691.

[43] J. Schulman, P. Moritz, S. Levine, M. Jordan, P. Abbeel, High-dimensional continuous control using generalized advantage estimation, 2015, http://dx.doi.org/10.48550/arXiv.1506.02438, arXiv preprint arXiv:1506.02438.

[44] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, M. Riedmiller, Deterministic policy gradient algorithms, in: International Conference on Machine Learning, Pmlr, 2014, pp. 387–395, URL https://proceedings.mlr.press/v32/silver14.html.

[45] T.P. Lillicrap, J.J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, Continuous control with deep reinforcement learning, 2015, http://dx.doi.org/10.48550/arXiv.1509.02971, arXiv preprint arXiv:1509.02971.

[46] 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, http://dx.doi.org/10.1016/j.energy.2020.117297.

[47] X. Zhang, X. Wang, P. Yuan, H. Tian, G. Shu, Optimizing hybrid electric vehicle coupling organic rankine cycle energy management strategy via deep reinforcement learning, Energy AI 17 (2024) 100392, http://dx.doi.org/10.1016/j.egyai.2024.100392.

[48] S. Dankwa, W. Zheng, Twin-delayed ddpg: A deep reinforcement learning technique to model a continuous movement of an intelligent robot agent, in: Proceedings of the 3rd International Conference on Vision, Image and Signal Processing, 2019, pp. 1–5, http://dx.doi.org/10.1145/3387168.3387199.

[49] R. Huang, H. He, X. Zhao, Y. Wang, M. Li, Battery health-aware and naturalistic data-driven energy management for hybrid electric bus based on TD3 deep reinforcement learning algorithm, Appl. Energy 321 (2022) 119353, http://dx.doi.org/10.1016/j.apenergy.2022.119353.

[50] J. Zhou, S. Xue, Y. Xue, Y. Liao, J. Liu, W. Zhao, A novel energy management strategy of hybrid electric vehicle via an improved TD3 deep reinforcement learning, Energy 224 (2021) 120118, http://dx.doi.org/10.1016/j.energy.2021.120118.

[51] T. Haarnoja, A. Zhou, P. Abbeel, S. Levine, Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, in: International Conference on Machine Learning, PMLR, 2018, pp. 1861–1870, http://dx.doi.org/10.48550/arXiv.1801.01290.

[52] Y. Lin, L. Chu, J. Hu, Z. Hou, J. Li, J. Jiang, Y. Zhang, Progress and summary of reinforcement learning on energy management of MPS-EV, Heliyon (2023) http://dx.doi.org/10.1016/j.heliyon.2023.e23014.

[53] 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) 108243, http://dx.doi.org/10.1016/j.est.2023.108243.

[54] W. Chen, J. Peng, J. Chen, J. Zhou, Z. Wei, C. Ma, Health-considered energy management strategy for fuel cell hybrid electric vehicle based on improved soft actor critic algorithm adopted with beta policy, Energy Convers. Manage. 292 (2023) 117362, http://dx.doi.org/10.1016/j.enconman.2023.117362.

[55] T. Li, W. Cui, N. Cui, Soft actor-critic algorithm-based energy management strategy for plug-in hybrid electric vehicle, World Electr. Veh. J. 13 (10) (2022) 193, http://dx.doi.org/10.3390/wevj13100193.

[56] L. Rolando, N. Campanelli, L. Tresca, L. Pulvirenti, F. Millo, Development of a Soft-Actor Critic Reinforcement Learning Algorithm for the Energy Management of a Hybrid Electric Vehicle, Tech. rep, SAE Technical Paper, 2024, http://dx.doi.org/10.4271/2024-37-0011.

[57] 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. Electrified Veh. 11 (14-11-01-0009) (2021) 113–132, http://dx.doi.org/10.4271/14-11-01-0009.

[58] J. Pavlovic, A. Tansini, J. Suarez, G. Fontaras, Influence of vehicle and battery ageing and driving modes on emissions and efficiency in plug-in hybrid vehicles, Energy Convers. Management: X 24 (2024) 100776, http://dx.doi.org/10.1016/j.ecmx.2024.100776.

[59] UNECE, Global technical regulations (GTRs), 2024, online documentation, https://unece.org/transport/standards/transport/vehicle-regulations-wp29/global-technical-regulations-gtrs.

[60] E. 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) ..., 2024, online documentation, https://tinyurl.com/53ecjp7s.

[61] S. Onori, L. Serrao, G. Rizzoni, Adaptive equivalent consumption minimization strategy for hybrid electric vehicles, in: Dynamic Systems and Control Conference, Vol. 44175, 2010, pp. 499–505, http://dx.doi.org/10.1115/DSCC2010-4211.

[62] L. Serrao, S. Onori, G. Rizzoni, ECMS as a realization of pontryagin's minimum principle for hev control, in: 2009 American Control Conference, IEEE, 2009, pp. 3964–3969, http://dx.doi.org/10.1109/ACC.2009.5160628.

[63] L.P. Kaelbling, M.L. Littman, A.W. Moore, Reinforcement learning: A survey, 1996, CoRR cs.AI/9605103, URL https://arxiv.org/abs/cs/9605103.
[64] N. Metropolis, S. Ulam, The Monte Carlo method, J. Am. Stat. Assoc. 44 (1949) 335, http://dx.doi.org/10.1080/01621459.1949.10483310.
[65] B. Efron, Bootstrap methods: Another look at the jackknife, Ann. Statist. 7 (1) (1979) 1–26, http://dx.doi.org/10.1214/aos/1176344552.
[66] D. Kingma, J. Ba, Adam: A method for stochastic optimization, in: International Conference on Learning Representations, ICLR, San Diego, CA, USA, 2015, http://dx.doi.org/10.48550/arXiv.1412.6980.
[67] O. Sundstrom, L. Guzzella, A generic dynamic programming matlab function, in: 2009 IEEE Control Applications, (CCA) & Intelligent Control, ISIC, 2009, pp. 1625–1630, http://dx.doi.org/10.1109/CCA.2009.5281131.
[68] K. Man, K. Tang, S. Kwong, Genetic algorithms: concepts and applications [in engineering design], IEEE Trans. Ind. Electron. 43 (5) (1996) 519–534, http://dx.doi.org/10.1109/41.538609.
[69] M.H. Beale, M.T. Hagan, H.B. Demuth, Deep learning toolbox, in: R2023b User's Guide, MathWorks, Inc, Natick, MA, USA, 2023.