Safe Reinforcement Learning for Energy Management of Electrified Vehicle with Novel Physics-Informed Exploration Strategy

Safe Reinforcement Learning for Energy Management of Electrified Vehicle with Novel Physics-Informed Exploration Strategy

Atriya Biswas*, *Member, IEEE,* Matteo Acquarone*, *Student Member, IEEE,* Hao Wang, *Student Member, IEEE,* Federico Miretti, *Member, IEEE,* Daniela Anna Misul, *Member, IEEE* and Ali Emadi, *Fellow, IEEE*

***Abstract*—This paper introduces a novel physics-informed exploration strategy for a deep reinforcement learning (DRL)-based energy management system (EMS), specifically targeting the challenge of dealing with constrained action sets. RL-based controllers for electrified vehicle energy management systems have faced obstacles stemming from the selection of infeasible actions, obstructing their practical deployment. The absence of a mechanism for assessing control action feasibility prior to application has compounded this issue, primarily due to the model-free nature of RL-based controllers. Adding a safety layer to the RL-based controller addresses the abovementioned issue, but this often results in suboptimal policies and necessitates an in-depth understanding of the powertrain. Alternatively, theoretical remedies incorporate penalty terms into the immediate reward function to manage infeasible conditions. However, this approach can slow down the training process as the agent learns to avoid infeasible actions. To surmount these challenges, this paper introduces a novel physics-informed exploration strategy, coupled with prioritized experience replay, enabling the agent to swiftly learn to avoid selecting infeasible control actions without the need for a separate safety layer. Real-time simulation results highlight the superior performance of the proposed DRL-based controller over the baseline DRL-based controller with a safety layer, particularly in terms of overall fuel consumption.**

***Index Terms*—Energy management system, failsafe policy, hybrid electric vehicle, Lagrange relaxation, prioritized experience replay, physics-informed exploration, safe reinforcement learning, real-time implementation, safety critic, safety layer.**

I. Introduction

The energy management strategy (EMS) is is the most crucial supervisory controller in an electrified vehicle from the energy economy perspective [1]. An EMS's task is to solve an electrified vehicle's energy management problem for a finite-length drive cycle, i.e., solving a finite-horizon nonlinear constrained optimization problem. Solving such an optimization problem is to optimally control one or multiple variables from the pool of powertrain variables, including but not limited to reference torque requests for powertrain prime movers, battery current, operating mode (all-electric mode or hybrid mode), and mechanical brake actuation, while satisfying the state and control variables' safety constraints. An EMS can be designed to fulfill various objectives, including but not limited to maximizing the energy economy of the powertrain [2], minimizing the tailpipe greenhouse gases (GHG) emissions [3], improving vehicle's drivability [4, 5], and improving battery longevity [6] for a given drive cycle or unknown drive cycle. In a real-world driving scenario, where the drive cycle data is unknown to the EMS, an intelligent control strategy for the EMS is pivotal for optimizing the powertrain's energy economy [7]. Academia has identified reinforcement learning (RL) algorithms as promising tools for designing an intelligent EMS for electrified vehicles [8].

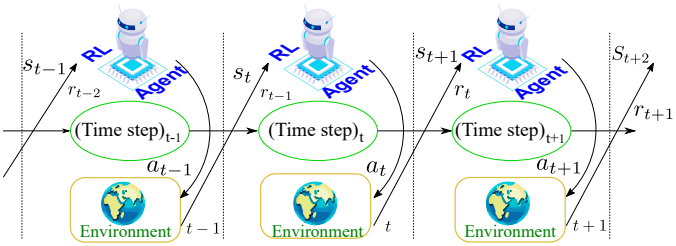

Fig. 1. Concept of finite-horizon sequential decision-making process with reinforcement learning agent.

RL algorithms advocate a controller, denoted as an "agent," to optimize long-term objectives rather than immediate ones within a sequential decision-making framework, as illustrated in Fig.1. The figure portrays the dynamic interaction between the environment and the RL agent in a finite-length Markov decision problem (MDP). In this interaction, the RL agent, having observed the state s_t of the environment at time "t," enacts an action a_t. Subsequently, the agent is bestowed with an immediate reward (r_t) corresponding to its executed action a_t in the ensuing time step. AlphaGo, the RL-agent computer that defeated the human world champion in the game of "Go," shook the entire world for the first time, disseminating the brilliance of RL algorithms to academic and industrial scholars [9]. Later in the last decade, scholars saw the advent of deep reinforcement learning (DRL) algorithms [10], a technological fusion between deep learning and RL, which outplayed traditional RL algorithms in several aspects, including feature representation, end-to-end learning, generalization, nonlinear function approximation, and complex policy representation, primarily due to DRL's ability to handle complex and high-

A. Biswas (Co-first and corresponding author), Hao Wang, and A. Emadi are with the McMaster Automotive Resource Center, McMaster University, Hamilton, ON, L8P0A6 Canada e-mail: (biswaa4@mcmaster.ca).

M. Acquarone (Co-first author), F. Miretti, and D. A. Misul are with Department of Energy (DENERG) and Center for Automotive Research and Sustainable Mobility (CARS), Politecnico di Torino, 10129 Torino, Italy.

This research is supported, in part, thanks to funding from the Natural Sciences and Engineering Research Council of Canada (NSERC), NSERC Industrial Research Chair in Electrified Powertrains, and Canada Research Chair in Transportation Electrification and Smart Mobility.

dimensional data. The prowess of DRL algorithms in sequential decision-making stormed through the internet when GT Sophy, a DRL-powered artificial intelligence (AI) agent, defeated the world's best gamers in a racing simulation game called Gran Turismo [11]. Early applications of RL in the mobility sector can be found as early as 2011, when it was applied to improve the energy efficiency of a human-driven hybrid electric bicycle [12]. In the same year, the application of DRL was introduced by R. Johri and Z. Filipi not only to improve the fuel economy of a series hydraulic hybrid vehicle but also to address the curse of dimensionality posed by dynamic programming dealing with discrete state and control variables [13].

Why do academics prefer RL or DRL algorithms over other control strategies (e.g., equivalent consumption minimization strategy (ECMS), heuristic, fuzzy rule-based, evolutionary algorithm, model predictive control (MPC), supervised learning, dynamic programming (DP)) for designing intelligent EMS in electrified vehicles? While each strategy has strengths and weaknesses, the suitability often depends on the application context. RL and DRL-based approaches outperform others in specific scenarios.

In contrast to strategies requiring a priori knowledge or predefined MDP models, RL/DRL-based methods operate independently, offering adaptability to changing models and drive cycles [14, 15]. Unlike many strategies demanding manual intervention for varying drive cycles and powertrain models, RL/DRL-based methods, once tuned, exhibit autonomy, enhancing their post-deployment applicability in commercial vehicle EMSs where manual tuning is impractical [16, 17]. Theoretically, RL/DRL-based strategies can autonomously learn and adapt their near-optimal control on-the-fly for changing driving scenarios post-deployment without server support [18]. This capability differs from over-the-air updates used by other EMS strategies through IoT devices and 5G networks [19]. Contemporary RL/DRL algorithms, unlike many traditional EMS strategies, allow real-time implementation during testing on an embedded microcontroller [20, 21] or through rapid prototyping in a hardware-in-the-loop (HIL) [22, 23]. Powered by nonlinear functional approximators like artificial neural networks, DRL algorithms effectively handle high-dimensional state-action spaces, suitable for high-fidelity vehicle environments [22, 24]. Lastly, unlike well-established strategies such as MPC, DP, and ECMS, which often require an accurate MDP model, RL and DRL algorithms may or may not need the MDP model. This flexibility enables them to operate in scenarios where obtaining an accurate MDP model is challenging, facilitating a model-free approach [25].

A. Background and Related Works

Despite the notable features of RL/DRL algorithms, two significant drawbacks—namely, the lack of safety [26] and AI explainability [27]—hinder their real-time implementation in actual hardware-based environments, both in industry and academia. While numerous real-time implementations have been reported, most remain confined to simulation environments [25]. Even when real-time implementations occur on HIL test benches, the interacting environments are often simulated on a computer [20, 21] or emulated using platforms like dSPACE, NI, and SpeedGoat [22, 23].

This article primarily addresses the lack of safety in DRL algorithms, with a secondary focus on AI explainability. Safety and feasibility of control actions are categorized into three levels based on the control policy's ability to conform to constraints, as illustrated in Fig.2 [26]. Level III safety ensures satisfaction of all constraints. In contrast, levels II and I guarantee probabilistic constraints and encourage constraint satisfaction, respectively. Traditional model-based strategies (DP, MPC, ECMS) can pre-evaluate conformity due to their inherent powertrain models, ensuring level III safety. In contrast, RL/DRL-based EMSs, with their model-free approach, cannot pre-evaluate such conformity, thereby not guaranteeing level III safety. The same model-free characteristic that grants RL/DRL algorithms advantages over traditional model-based strategies also impedes their successful implementation in hardware-based environments, placing them behind model-based strategies in hardware implementation. Despite the control search space constraints in RL/DRL algorithms, random exploration in the early learning phase may lead to occasional violations of state variable constraints, transitioning into unsafe or infeasible environment states.

In the pursuit of aligning RL/DRL algorithms with safety and feasibility constraints, prevalent approaches can be delineated into three categories:

1) Employing the Lagrange relaxation method [28]
2) Incorporating a safety critic
3) Shielding the RL/DRL agent with a safety layer

The first two methods demonstrate efficacy in reducing the instances of unsafe control outputs from the RL/DRL agent over the learning progression, culminating in heightened safety by the end of the learning phase. Conversely, the third method consistently assures the safety of the environment.

Using Lagrange multipliers is a promising avenue, leveraging their capability to transform a constrained policy optimization problem into an unconstrained one. An example of a constrained optimization problem is shown in Eq.1.

$$\max_{x_i \in \mathbb{X}} J(x_i) \ , \ s.t., \qquad f(x_i) \leq \alpha_j \quad for\ all\ time\ steps \tag{1}$$

This constrained optimization problem can be expressed as an unconstrained one, as shown in Eq.2, using Lagrange multiplier ($\mathcal{L}$).

$$\min_{\lambda_j \geq 0} \max_{x_i \in \mathbb{X}} \mathcal{L}(\lambda_j, x_i) = \min_{\lambda_j \geq 0} \max_{x_i \in \mathbb{X}} \Big[J(x_i) - \sum \lambda_j \big(f(x_i) - \alpha_j \big) \Big] \tag{2}$$

The solution of the unconstrained optimization problem, i.e., the optimal value of λ_j and x_i can be obtained by minimizing the gradient of the Lagrange in Eq.2 with respect to λ and x, respectively, as shown in Eq.3.

$$\nabla_x \mathcal{L}(\lambda_j, x_i) = 0; \text{ and } \nabla_\lambda \mathcal{L}(\lambda_j, x_i) = 0 \tag{3}$$

This makes the Lagrange relaxation method an attractive solution for RL/DRL algorithms seeking to adhere to safety constraints in several constrained optimization problems in transportation electrification, such as optimal electric vehicle

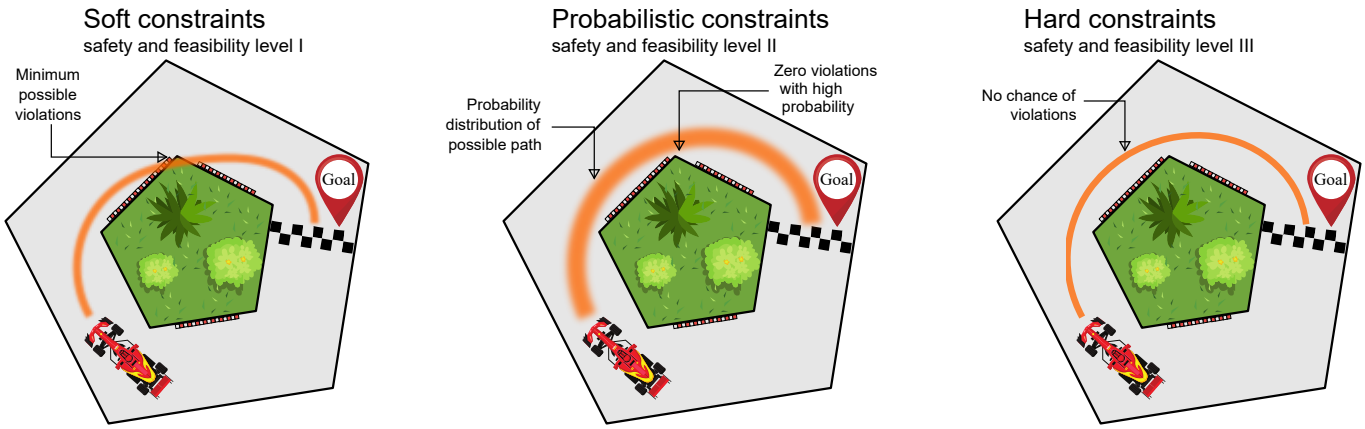

Fig. 2. Conceptual diagram explaining the difference among three levels of safety and feasibility of control policy.

(EV) charge scheduling [29, 30, 31] and optimal energy management of hybrid electric vehicle (HEV)s [32]. Despite its promise, this approach introduces complexities by converting the optimization problem into a multi-objective one [33]. Additionally, articulating the precise nonlinear transfer functions between the control policy and each state variable becomes challenging, particularly in scenarios with a high number of state variables. This often compels Lagrange designers to use a common penalty term for all safety constraint violations [30, 32]. Despite the ingenuity of this approach, it diminishes the optimizer's flexibility to adjust the control policy for individual constraint satisfaction, resulting in sub-optimal performance. Using a common penalty term introduces another challenge. When dealing with multiple safety constraints, it hampers the designer's ability to discern how a policy update enhancesRL/DRL conformity to specific safety constraints. This lack of clarity makes RL/DRL algorithms more susceptible to skepticism regarding their AI explainability.

An alternative strategy involves employing a safety critic, which is particularly advantageous when state safety constraints are unknown. The safety critic assesses the degree of unsafety associated with a specific state using a safety value function, denoted as $V_c(s)$, or evaluates the unsafety of taking a particular action at a given state using a safety quality function, denoted as $Q_c(s, a)$ [34]. Both these safety functions are evaluated and updated iteratively during the policy iteration step [35]. The safety critic tends to overestimate these functions in the early learning phase, ensuring conservative decision-making for enhanced safety [36]. As learning progresses, improved estimates allow the agent to explore broader areas of the action space [37]. However, this approach may have prolonged convergence times in DRL algorithms, making it less suitable for automotive applications where safety constraints are generally known.

When safety constraints are known, employing a safety layer with a failsafe policy ($\pi_{failsafe}$) in RL/DRL-based EMS is advisable, especially during training, to prevent unsafe control actions [32]. While not obligatory, it is common practice to retain the safety layer during the testing phase to ensure the utmost safety of the environment. If post-execution verification deems them unsafe, the safety layer replaces RL/DRL-based EMS control actions with predetermined safe controls [38]. Safety layers consist of simple and conservative rules derived from standard practices or expert knowledge [39]. However, it's essential to note that crafting an infallible safety layer necessitates a robust understanding of the specific electrified powertrain domain [40]. Moreover, the conservative rules inherent in the safety layer can potentially influence the learning of the RL/DRL algorithm in a sub-optimal manner [41, 42]. Notably, without rigorous domain knowledge, the safety layer can be curated based on human feedback [38].

B. Novelty and Contribution

Intrinsic shortcomings and avenues for enhancement in current approaches addressing the safety of DRL/RL-based EMS include:

- Using multiple Lagrange multipliers may result in sub-optimal EMS performance, and the shared multiplier raises concerns about AI explainability.
- Neither safety critics nor the Lagrange relaxation method can guarantee level III safety during training and testing.
- Failsafe policies ($\pi_{failsafe}$) contribute to sub-optimal EMS performance. Moreover, existing literature lacks studies quantifying the degradation in EMS performance when employing failsafe policies compared to other safe reinforcement learning approaches.

In light of these drawbacks and motivated by the potential enhancement of DRL-based EMS policy safety through human feedback, this article proposes a novel physics-informed exploration (PIE) strategy. The primary algorithm in the DRL framework is the twin delayed deep deterministic policy gradient (TD3) method, and the proposed PIE strategy is employed inside the TD3 method. The key contributions of this study are summarized as follows:

1) A PIE strategy for DRL-based EMS, which improves adherence to safety constraints and reduces overall violations throughout training episodes, is proposed.
2) Targeted area-based control policy modifications are implemented based on identified reasons for safety constraint violations, enhancing the AI explainability of the proposed strategy through feedback from the environment.

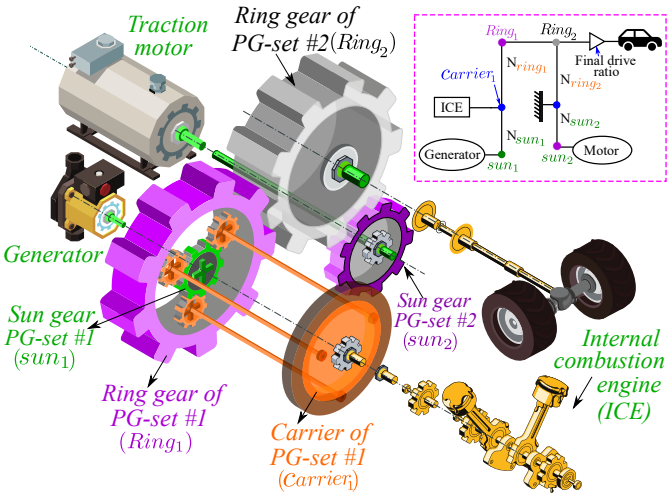

Fig. 3. Schematic and lever diagram of the third generation Toyota hybrid electric propulsion system.

3) Integration of the PIE strategy with prioritized experience replay (PER) in the overall DRL framework, achieving accelerated convergence and adherence to constraints, thereby reducing overall training time.
4) The superiority of the proposed PIE-based DRL framework over a conventional safe DRL approach, which includes penalized rewards and failsafe policies, is demonstrated through simulations. A separate DRL framework with a failsafe policy is also presented for comparative analysis.

The remainder of the paper is organized as follows: Section II will outline the mathematical and Simulink®-based modeling of the third generation Toyota hybrid system (Toyota hybrid system (THS)). Section III will delineate the fundamentals of the RL framework employed for solving the energy management problem. This section also presents the baseline RL framework encapsulated with a safety layer. Section IV elaborates on the TD3 algorithm and how PER helps it to achieve expedited convergence. Section V introduces the novel PIE and explains how it satisfies all the constraints of the energy management problem without any safety layer. Section VI presents a few critical results from the Co-simulation framework between Simulink® and Python, and subsequently, conclusions are drawn in section VII.

II. Mathematical and Simulink® Modeling of HEV Powertrain

A. Modeling of dynamics for different modes

The third generation THS, depicted in Fig.3, has been selected as the hybrid electric propulsion system (HePS) architecture for this study. The most important components of the HePS are an internal combustion engine (ICE), two electric motors, a battery, a planetary gear-set, and a speed reduction gear mesh. The authors have chosen the parameters and maps of the components so that the vehicle plant can be modeled as close as possible to 2010 Toyota Prius. The third generation THS can facilitate two distinct electrified modes, i.e., a pure electric mode (PEM) and a hybrid electric mode (HEM). The specifications of the main components of the third generation THS are provided in Tab.I.

B. Inertia-based powertrain dynamics modeling

This subsection will formulate the inertia-based powertrain dynamics of the third generation THS. As depicted in Fig. 3, the ICE is attached to the planet carrier of the planetary gear-set. The generator is attached directly to the sun gear, and the traction motor is attached to the ring gear through a speed reduction gear and a counter-driven gear. The speed reduction gear mesh is equivalent to a planetary gear-set with its carrier being grounded, as depicted at the top right corner of Fig. 3. The inertia-based powertrain dynamics of the PEM and HEM modes can be expressed through the method described in [4, 43] as follows:

$$\tau_{\text{out}} - J_{\text{out,eq.}}\ddot{\theta}_{\text{out}} + \left(\frac{\beta_1}{\beta_1+1}\right)\left(\tau_{\text{ICE}} - J_{\text{ICE,eq.}}\ddot{\theta}_{\text{ICE}}\right) + \beta_2\left(J_{\text{mot,eq.}}\ddot{\theta}_{\text{mot}} - \tau_{\text{mot}}\right) \tag{4}$$

$$\tau_{\text{ICE}} - J_{\text{ICE,eq.}}\ddot{\theta}_{\text{ICE}} = \left(\beta_1+1\right)\left(J_{\text{gen,eq.}}\ddot{\theta}_{\text{gen}} - \tau_{\text{gen}}\right) \tag{5}$$

$$\ddot{\theta}_{\text{ICE}}\left(\beta_1+1\right) = \beta_1\ddot{\theta}_{\text{out}} + \ddot{\theta}_{\text{gen}} \tag{6}$$

$$\ddot{\theta}_{\text{mot}} = -\beta_2\ddot{\theta}_{\text{out}}, \tag{7}$$

where τ_{out}, τ_{ICE} and τ_{mot} are the transmission output, engine, and motor torques, respectively; τ_{out}, $\ddot{\theta}_{\text{ICE}}$, and $\ddot{\theta}_{\text{mot}}$ are the angular accelerations of output ring, ICE, and motor, respectively; β_1 and β_2 are the gear ratios of the planetary gear-set and speed reduction gear mesh, respectively. The ICE activation is the only differentiating factor between PEM and HEM. In PEM, the traction motor solely satisfies the driver's power demand and the generator rotates freely without applying torque to the transmission output. On the other hand, in HEM the traction motor and ICE satisfy the power demand. The generator is critical in transitioning from PEM to HEM, since it starts applying positive torque on the sun gear of the first planetary gear-set as it receives the command of PEM to HEM transitioning. While the generator cranks the ICE, a dynamic motor-torque compensation control enables the traction motor to satisfy the torque demand at the transmission output and alleviate the powertrain instability resulting from ICE's torque ripple [44]. The equivalent inertia $J_{\text{out,eq.}}$ can be further disintegrated into rudimentary elements:

$$J_{\text{out,eq.}} = \left\{\frac{J_{\text{veh}}}{i_{\text{fd}}^2} + J_{\text{diff}}\right\} + J_{\text{ring}}, \tag{8}$$

where the J_{veh}, J_{diff}, and J_{ring} are the vehicle equivalent inertia, the differential inertia, and ring inertia respectively, and i_{fd} is the final drive ratio. Torque τ_{out} and power P_{out} at the transmission output are computed from road loads, gravitational load, inertial load, aerodynamic load, and torque

losses τ_{loss} at the differential, due to gear meshing, and gear spinning:

$$\tau_{\text{out}} = \frac{\left(m_{\text{veh}} g \sin(\vartheta) + m_{\text{veh}} g \cos(\vartheta)(c_{r,1} v_{\text{veh}} + c_{r,2})\right) r_{\text{wh}}}{i_{\text{fd}}} + \frac{\left(m_{\text{veh}} g(\frac{\partial v_{\text{veh}}}{\partial t}) + \frac{1}{2} c_d \rho A_f v_{\text{veh}}^2 + \tau_{\text{brake}}\right) r_{\text{wh}}}{i_{\text{fd}}} + \tau_{\text{loss}} \tag{9}$$

$$P_{\text{out}} = \tau_{\text{out}} \omega_{\text{out}}. \tag{10}$$

Here, $c_{r,1}$ and $c_{r,2}$ are rolling resistance coefficents, r_{wh} is the wheel radius, c_d is the aerodynamic drag coefficent, ρ is the air density, and A_f is the vehicle's frontal area.

C. ICE modeling

A map-based 1.8 L ICE model is modeled through Wide open throttle (WOT) torque and engine efficiency (η_{ICE}) maps, shown in Fig.4. The ICE's fuel consumption rate can be calculated from the efficiency map as a function of ICE angular velocity ω_{ICE}, and torque τ_{ICE}:

$$\begin{aligned} \eta_{\text{ICE}} &= f(\omega_{\text{ICE}}, \tau_{\text{ICE}}) \\ \dot{m}_{fuel} &= \frac{\omega_{\text{ICE}} \cdot \tau_{\text{ICE}}}{\eta_{\text{ICE}} \cdot H_l}, \end{aligned} \tag{11}$$

where H_l is the fuel's lower heating value.

D. Electric machine modeling

Both traction motor and generator are modeled through maximum torque $\tau_{\text{mot/gen,max}}$ curve and efficiency $\eta_{\text{mot/gen}}$ maps as functions of motor and generator speed $\omega_{\text{mot/gen}}$ and torque $\tau_{\text{mot/gen}}$:

$$\begin{aligned} \tau_{\text{mot/gen,max}} &= f(\omega_{\text{mot/gen}}) \\ \eta_{\text{mot/gen}} &= f(\omega_{\text{mot/gen}}, \tau_{\text{mot/gen}}) \end{aligned} \tag{12}$$

Although maps for the traction motor are widely available in the literature [45, 46], the correct generator maps for the third-generation THS are not available to the best of the author's knowledge. Hence, tentative generator maps, shown in Fig.5, are used in this article.

E. High-voltage battery modeling

The battery pack comprises 168 Panasonic Nickel-Metal cells, each having a nominal voltage of 1.2V and 6.5Ah capacity [47, 46]. The battery is modeled through an equivalent circuit model, characterized by temperature and SOC-dependant open circuit voltage (V_{oc}) curves and internal resistance (Ω_{batt}) curves. The state-of-charge (SOC) dynamics can be written through the following equation:

$$\dot{SOC}(t) = -\frac{I_{\text{batt}}}{C_{\text{batt}}} = -\frac{V_{\text{oc}} - \sqrt{V_{\text{oc}}^2 - 4 P_{\text{batt}} \cdot \Omega_{\text{batt}}}}{2 \Omega_{\text{batt}} \cdot C_{\text{batt}}}, \tag{13}$$

where V_{oc} and Ω_{batt} are functions of the previous time-step's SOC value, and P_{batt} is derived from the following equation:

$$P_{\text{batt}} = \frac{P_{\text{out}} - P_{\text{ICE}}}{\eta_{\text{batt}}^{\text{sign}(I_{\text{batt}})}} \tag{14}$$

TABLE I
SPECIFICATION OF THE VEHICLE AND MAIN COMPONENTS

Component	Parameter	Value
Vehicle	Mass m_{veh} /Inertia J_{veh}	1530 kg/154 kg $\cdot$ m^2
	Wheel radius r_{wh}	0.3173 m
	Height/ Width	1.75 m/ 1.48 m
Internal combustion engine (ICE)	Inertia J_{ICE}	0.1544 kg $\cdot$ m^2
	Max. speed $\omega_{\text{ICE,max}}$	5500 rpm
	Max. torque $\tau_{\text{ICE,max}}$	142 Nm at 4050 rpm
	Volume displacement	4 Cyl., 1.8 L
	Max. power	73 kW at 5175 rpm
Traction motor	Inertia J_{mot}	0.0226 kg $\cdot$ m^2 ,
	Max. speed, Max. Volt.	13500 rpm, 240 V
	Max. torque $\tau_{\text{mot,max}}$	200 Nm at 2000 rpm
	Max. power	60 kW at 2240 rpm
Generator	Inertia J_{gen}	0.001 kg $\cdot$ m^2
	Max. speed, Max. Volt.	13500 rpm, 240 V
	Max. torque $\tau_{\text{gen,max}}$	40.8 Nm at 7615 rpm
	Max. power	42 kW at 7615 rpm
High voltage battery	Max. capacity/ Nom. Voltage	6.5 Ah/ 201.6 V
	Max. power	27 kW

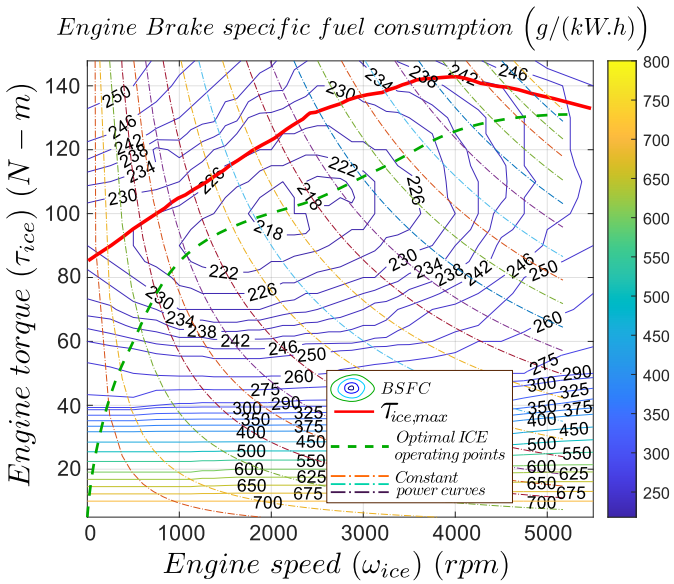

Fig. 4. Brake specific fuel consumption (BSFC) map, Wide open throttle torque (WOT) w.r.t engine speed, constant power curves, and optimal engine operating line for the 1.8 L. Atkinson engine used in third generation THS.

As is typically the case with HEVs [48], the battery SOC must be kept inside the operational range $[SOC_{low}, SOC_{up}]$, SOC_{low}=0.2, and SOC_{up}=0.6 in this work.

III. RL Framework for Solving Energy Management Problem

In this section, some basic concepts of RL algorithms are illustrated. RL is a machine learning branch in which the agent learns a near-optimal policy by constantly interacting with the surrounding environment through a trial-and-error process. The agent's main objective is to maximize a particular metric of performance, the *discounted return*.

Since the RL agent can receive a partial subset of the state ($\mathcal{S}$) of the system, named *observation*, the interaction between the RL agent and the environment is modeled through a partially observable Markovian decision process (POMDP)

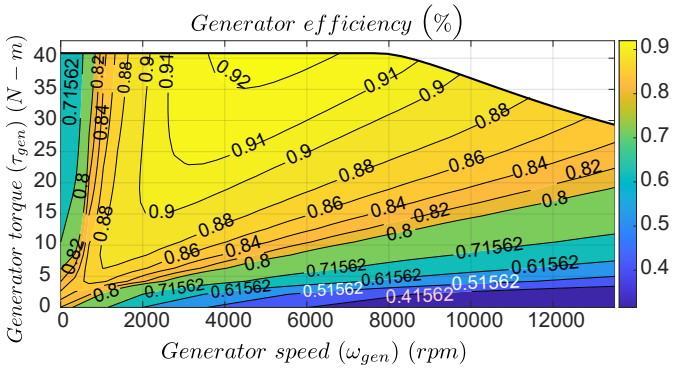

Fig. 5. Efficiency map of the generator used in modeling the powertrain which closely resembles the third generation THS.

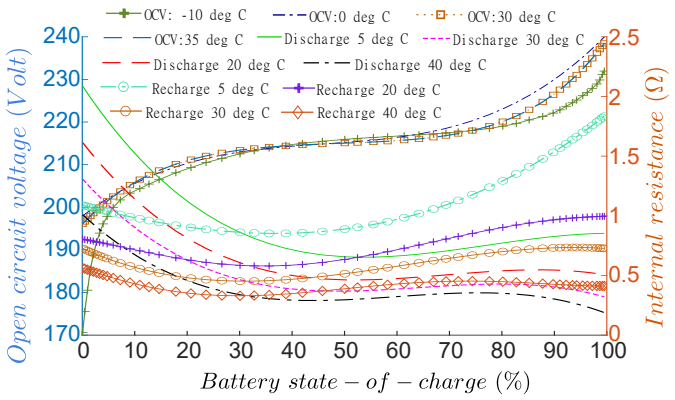

Fig. 6. SOC-V_{oc} curves and internal resistance curves of the high-voltage battery pack, at different temperatures, used in the powertrain modeling in this article.

[49]. At each time step, the RL agent receives the observation o_t from the environment and selects an action a_t. The environment is affected by the action a_t and is characterized by a new state s_{t+1} in the next time instant; in the meanwhile, the agent receives the reward r_t as a consequence of the action a_t. For simplicity, from now on, the observations are referred to as state s_t. The objective of any RL agent is to maximize the discounted sum of the rewards, also known as the discounted return $\mathcal{G}_t$, obtained throughout an episode:

$$\mathcal{G}_t = \sum_{k=t+1}^{T} \gamma^{k-t-1} r_k \tag{15}$$

, where γ is the discount factor, and T is the final time step. Several RL agents try to estimate the value of the expected discounted return (Q-value) to properly select the action a_t at a given state s_t. In other words, the Q-value of s_t and a_t is referred to as the agent's estimate of the expected discounted return assuming that the agent selects the action a_t for the environment characterized by state s_t and then follows the best policy.

In an RL framework several quantities must be carefully selected to correctly set the problem. In particular, the set of actions, states, and the reward function must be defined. Since the main purpose of our controller is to produce an optimal energy management strategy, we selected the ICE power $a = \{P_{\text{ICE}}\}$ as the continuous action variable. Once the power requested to the ICE is selected, the rotational speed and torque are fully defined by the ICE optimal operating line, as shown in Fig. 4. In this work, the observation set $s = \{P_{\text{out}}, SOC, v_{\text{veh}}, P_{\text{ICE,p}}\}$ is composed of the power requested at the transmission output, the SOC, the vehicle's current velocity, and the power requested to the ICE in the previous time-step ($P_{\text{ICE,p}}$), allowing the agent to receive every necessary information to select the optimal action.

Another important quantity that must be defined is the reward function. Since the main objective of a RL algorithm is to maximize the sum of rewards accumulated over the episode, the reward function must be deeply connected with the physical objective that we want to achieve. The main objectives of this study are to minimize fuel consumption, guarantee battery charge sustaining, and avoid infeasible conditions. In this work, two reward functions of different complexity are developed to test the robustness of the proposed RL agents with different reward objectives.

A first simpler reward function, referred to as SOC-oriented [50], is formulated to directly tackle the problem of avoiding infeasibilities while still maintaining battery charge sustaining. Since fuel consumption minimization is completely disregarded, the optimal policy is easier to learn. Regarding the formulation of the reward function, the r_{SOC} term of the function aims at maintaining the SOC as close as possible to the final desired SOC SOC_{ref}, while a penalty term r_{inf} is added to the reward if an infeasible condition is encountered:

$$r_{SOC} = a_{SOC} - b_{SOC}\left(\frac{SOC - SOC_{ref}}{SOC_{low} - SOC_{ref}}\right)^2 \tag{16}$$

$$r_{\text{inf}} = -1 \tag{17}$$

$$r = \begin{cases} r_{SOC} & \text{if } u \text{ is feasible} \\ r_{\text{inf}} & \text{if } u \text{ is infeasible} \end{cases} \tag{18}$$

, where a_{SOC} and b_{SOC} are two tuning coefficients. In order to normalize the reward between -1 and 1, these coefficients were set to $a_{SOC} = 1$ and $b_{SOC} = 2$. It is important to specify that if an infeasibility is encountered, the episode is stopped as it would happen in real-world scenarios.

The second reward function, referred to as FC-oriented [50], is more complex since it aims at achieving the minimization of fuel consumption, still maintaining the final SOC near the reference value, and avoiding infeasibilities. This reward function keeps the penalty r_{inf} and computes the weighted average between the term r_{SOC} and a new reward term r_{fuel} added to reduce the fuel consumption:

$$r_{\text{fuel}} = a_{\text{fuel}} - b_{\text{fuel}}\left(\frac{m_{\text{f}}}{m_{\text{f,max}}}\right)^2 \tag{19}$$

$$r = \begin{cases} \frac{(c_{SOC} \times r_{SOC} + c_{\text{fuel}} \times r_{\text{fuel}})}{(c_{SOC} + c_{\text{fuel}})} & \text{if } u \text{ is feasible} \\ r_{\text{inf}} & \text{if } u \text{ is infeasible} \end{cases}. \tag{20}$$

Here, m_{f} is the fuel mass consumed in a time step and $m_{\text{f,max}}$ is the maximum fuel mass that can be burned by the ICE in a time step; a_{fuel} and b_{fuel} are two tuning parameters that are set to 1 and 2, respectively, to normalize the reward between

TABLE II
LIST OF HYPERPARAMETERS USED IN REWARD FUNCTIONS

Reward hyper-parameters	a_{SOC}, a_{fuel}	b_{SOC}, b_{fuel}	c_{SOC}	c_{fuel}	$m_{f,max}$(g)	SOC_{low}, SOC_{up}	SOC_{ref}
Value	1	2	1	2	3	0.2, 0.6	0.4

-1 and 1; c_{fuel} and c_{SOC} are weighting coefficients. The normalization of r_{SOC} and r_{fuel} reward terms between -1 and 1 makes it possible to compare them even if they are related to different physical quantities.

As results will show in Section VI, the addition of a term in the reward formulation complicates the training process [24]. The tuning factors and weighting coefficients used in formulating the reward functions are tabulated in Tab.II.

A. Safety layer

In accordance with the introductory discourse, it is evident that RL-based EMS without safety layer may not guarantee the attainment of Level III or Level II safety standards for the selected control strategies during the training phase. Indeed, the adopted control policies might breach safety criteria multiple times during the training phase, thereby posing potential risks to physical components or simulation testbeds. In light of these considerations regarding the safety assurance capabilities of RL-based EMSs without safety layer, the implementation of a safety layer becomes imperative should expensive simulated environments be utilized in both the training and testing phases. Nonetheless, it has become a prevalent best practice to wrap RL-based EMSs in a safety layer, even when employing low-cost and low-risk simulated environments for either training or testing purposes.

However, the implementation of the safety layer can generally lead to erroneous training of the RL agent, consequently resulting in sub-optimal performance. While the subsection III-A1 provides explanations of how the safety layer works in the RL framework, the second subsection III-A2 provides a theoretical rationale substantiating the suboptimal performance of a RL agent provided with a safety layer.

1) Safe RL framework with safety layer: An RL agent must adhere to the subsequent safety constraints when formulating the control policy for the energy management problem outlined in this study:

$$P_{\text{ICE}} \leq P_{\text{ICE,max}}, \tag{21}$$
$$P_{\text{ICE}} \geq P_{\text{ICE,min}}, \tag{22}$$
$$P_{\text{gen}} \leq P_{\text{gen,max}}, \tag{23}$$
$$P_{\text{gen}} \geq P_{gen_{min}}, \tag{24}$$
$$P_{\text{mot}} \leq P_{\text{mot,max}}, \tag{25}$$
$$P_{\text{mot}} \geq P_{\text{mot,min}}, \tag{26}$$
$$P_{\text{batt}} \leq P_{\text{batt,max}} \wedge I_{\text{batt}} \leq I_{\text{batt,max}}, \tag{27}$$
$$P_{\text{batt}} \geq P_{\text{batt,min}} \wedge I_{\text{batt}} \leq I_{\text{batt,min}}, \tag{28}$$
$$SOC_{\min} \leq SOC \leq SOC_{\max}. \tag{29}$$

where $\tau_{\text{ICE,max}}$ is the wide-open throttle torque corresponding to the requested ω_{ICE}. Therefore, all the possible infeasible conditions, excluding the ones related to the state constraints, are:

$$f_{\text{ICE,ub}} = P_{\text{ICE}} > P_{\text{ICE,max}}, \tag{30}$$
$$f_{\text{ICE,lb}} = P_{\text{ICE}} < P_{\text{ICE,min}}, \tag{31}$$
$$f_{\text{gen,ub}} = P_{\text{gen}} > P_{\text{gen,max}}, \tag{32}$$
$$f_{\text{gen,lb}} = P_{\text{gen}} < P_{gen_{min}}, \tag{33}$$
$$f_{\text{mot,ub}} = P_{\text{mot}} > P_{\text{mot,max}}, \tag{34}$$
$$f_{\text{mot,lb}} = P_{\text{mot}} < P_{\text{mot,max}}, \tag{35}$$
$$f_{\text{batt,ub}} = P_{\text{batt}} > P_{\text{batt,max}} \vee I_{\text{batt}} > I_{\text{batt,max}}, \tag{36}$$
$$f_{\text{batt,lb}} = P_{\text{batt}} < P_{\text{batt,min}} \vee I_{\text{batt}} < I_{\text{batt,min}}. \tag{37}$$

The RL-based framework with a safety layer is shown in Fig. 7. When the output of the RL-based EMS adheres to the predefined safety constraints, it is immediately executed in the environment model. Conversely, if the output fails to meet these safety criteria, a feasible action originating from the safety layer supersedes the RL-based EMS's output and is executed within the environment. The pseudocode of the algorithm responsible for prescribing safe actions at specific states is presented in Algorithm 1. To enhance readability, the symbol $f_{P_{ICE},low}$ refers to one or more infeasibilities between $f_{\text{ICE,lb}}$, $f_{\text{gen,ub}}$, $f_{\text{mot,ub}}$, or $f_{\text{batt,ub}}$, while $f_{P_{ICE},high}$ refers to $f_{\text{ICE,ub}}$, $f_{\text{gen,lb}}$, $f_{\text{mot,lb}}$, or $f_{\text{batt,lb}}$.

Algorithm 1 Safety layer algorithm pseudo code

```
while n < N do                                   ▷ n: episode number
  while t < T do                                 ▷ T: final time of driving cycle
    Select action a from state s_t through actor network
    Add gaussian noise 𝒩 to the action a for exploration
    Compute P_ICE from the selected action a + 𝒩
    Initialize: P_ICE,eval = P_ICE
    if P_ICE,eval led to infeasibility then
      Initialize: f_inf = 1 ▷ f_inf is the infeasibility flag
      Initialize: iteration = 1
      while f_inf = 1 do
        if P_ICE,eval led to f_{P_ICE,low} then
          P_ICE,eval = P_ICE + (iteration · 1kW)
          iteration = iteration + 1
        else if P_ICE,eval led to f_{P_ICE,high} then
          P_ICE,eval = P_ICE − (iteration · 1kW)
          iteration = iteration + 1
        else
          f_inf=0
        end if
      end while
    end if
    Execute P_ICE,eval
  end while
end while
```

2) Effect of safety layer on the Training of an RL agent: In all the algorithms tested in this work, the action is bounded within a set $\mathcal{A}^*$, regardless of the state. This set is simply

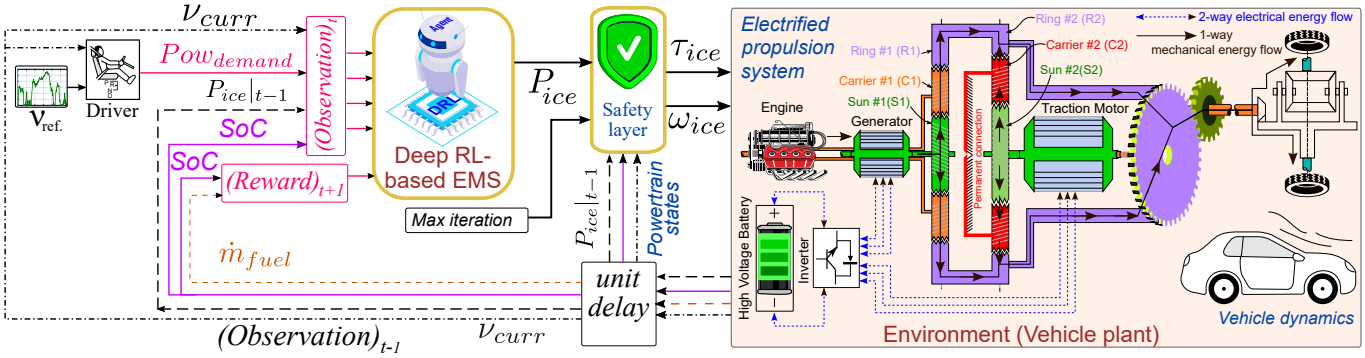

Fig. 7. Schematic of the EMS framework outputting a near-optimal feasible control for the powertrain through an RL-based framework and a safety layer.

bounded by -1 and +1[1]. For a given state s_t at time t, we define $\mathcal{A}(s_t)$ as the set of feasible actions, which is a subset of $\mathcal{A}^*$. For each time t, we assume that $\mathcal{A}(s_t)$ is restricted by a lower bound a_{lb} and an upper bound a_{ub}. These bounds are determined by the powertrain constraints written in this subsection.

When using the safety layer, the agent can select any action of the set $\mathcal{A}^*$, and the safety layer automatically corrects the action to the nearest feasible action, which is either a_{lb} or a_{ub}, and the simulation is not stopped. As a result, the agent receives a reward which does not reflect the infeasibilty of the action and it is unable to learn how to avoid that action in the future. Due to the use of the safety layer, feasible action is always guaranteed for the environment. However, the transition stored in the memory buffer, i.e., $< s_t, a_t, r, s_{t+1} >$, contains the agent's infeasible action along with the reward r which is not penalized. The agent will use this transition information later to update the actor and critic nets. In this case, there is a substantial risk that the agent will be stuck choosing actions from the infeasible set since the reward r is not affected by the infeasibility penalty. Thus, the standard exploration strategy used with the TD3 does not guarantee an exploration of the actions inside the feasible set, and the true optimal action could never be selected by the agent.

IV. Twin Delayed Deep Deterministic Policy Gradient Agent-based RL Framework

Some of the most popular RL algorithms, such as tabular Q-learning, deep Q-network (DQN) and double deep Q-network (DDQN), can only handle discrete action spaces. The discretization error arising from finite resolution in both the state and action variables spaces limits the ability of such algorithms to find the true optimal solution [2]. Therefore, the TD3 algorithm is selected in this work to avoid state and action variables discretization. TD3 is one of the state of the art off-policy actor-critic algorithms and is considered an enhanced version of the Deep Deterministic Policy Gradient (DDPG). Indeed, TD3 tackles several DDPG disadvantages, introducing *Clipped Double Q-Learning* to address the DDPG overestimation bias problem [51] through the use of two critic networks, *Target Policy Smoothing Regularization* to reduce the variance in the target policy [52], and a delayed update of the policy and target nets to give more time to the critic NN to get Q-values closer to the target.

TD3 is categorized as an off-policy actor-critic algorithm that handles continuous state and action spaces. Actor-critic algorithms use two distinct kinds of neural networks to learn the optimal policy: actor μ and critic Q nets. The actor specifies the current policy by deterministically mapping the observations to a specific action, while the critic estimates the Q-values. One of the main peculiarities of the TD3 algorithm is the use of a pair of independently trained critic networks Q_1 and Q_2, relying on Double Q-learning. Moreover, three additional target networks, i.e., one target actor μ_t and two target critic $Q_{t,1}$ and $Q_{t,2}$, are used for stability reasons to update the neural network (NN) weights during training. At each time step of the training phase, an action a_t is selected using the actor based on the state s_t and executed after adding a random noise function to the action. The resulting state s_{t+1} is observed and the reward r is collected; the transition, i.e., the set of values $< s_t, a_t, r, s_t >$, is stored in the *experience memory buffer*. Moreover, n transitions are sampled from the experience replay buffer, and are used to update the actor and critic networks through the respective cost functions:

$$
\begin{aligned}
L_a &= \frac{1}{n}\sum_{i=1}^{n} Q\big(s, \mu(s)\big) \\
L_{c,1/2} &= \frac{1}{n}\sum_{i=1}^{n} (TD_{error,1/2})^2 \\
TD_{error,1/2} &= y - Q_{1/2}\big(s, a|\theta^Q_{1/2}\big) \\
y &= r + \gamma \min_i Q_{t,i}\big(\theta^{Q_{t,i}}\big)
\end{aligned}
\tag{38}
$$

where y is the target value, and the notation θ^x refers to the weights of the x neural network. The term TD_{error} is defined as the temporal difference (TD) error, a measure of how close the target value y computed through the target networks is to the one $Q\big(s, a|\theta^Q\big)$ computed by the critic. As previously mentioned, the use of two critics allows to reduce the overestimation problem of the Q-values. Indeed, by taking the minimum between the two estimates $\min_i Q_{t,i}\big(\theta^{Q_{t,i}}\big)$ from the target networks, the value target cannot introduce any additional overestimation over using the standard Q-learning target $Q_{t,i}\big(\theta^{Q_{t,i}}\big)$. The main disadvantages of TD3 are the possible underestimation of the Q-values due to this update

[1]Since the actions are normalized between -1 and 1, $a = -1$ corresponds to $P_{\text{ICE}} = P_{\text{ICE,min}}$ and $a = -1$ corresponds to $P_{\text{ICE}} = P_{\text{ICE,max}}$

rule, and the higher computational time than DDPG due to the higher number of critic networks to train. The target nets weights are updated through a soft target update as shown in [51].

The standard TD3 exploration strategy adds a random noise $\mathcal{N}(0, \epsilon)$ function to the action output of the actor net. Although this exploration strategy has proven to be effective in different applications, it presents some notable limitations regarding the avoidance of infeasible actions. This article will demonstrate how the proposed novel physics-guided exploration strategy outperforms the standard exploration strategy in this regard.

A. Experience replay

In TD3 agent, the experience replay memory greatly improves the sample efficiency of the algorithm by enabling data to be reused multiple times for training, instead of throwing away data immediately after collection, and enhances the stability of the network during training.

1) General idea of Experience Replay Memory: The standard experience replay is a memory buffer with a fixed capacity of storing a maximum of N (in this paper $N = 50000$) transitions ($< s_t, a_t, r, s_{t+1} >$). It is implemented as a circular buffer, i.e., the oldest transition in the buffer is removed to welcome the newest transition. During the training phase, a batch of n transitions (in this paper $n = 32$) are randomly sampled (with uniform distribution) to train the actor and the critic NNs.

2) Prioritized Experience Replay (PER): PER [53] helps the TD3 agent learn to avoid infeasibilities more quickly than uniform sampling of the standard experience replay.

With PER, the transitions i to be used for training are selected based on sampling priorities p_i, defined as the value of the temporal difference error. Since the TD-error is an approximate measure of how unexpected (and therefore informative) the experience provided by that transition, PER increases sampling efficiency. The sampling probability for each transition is evaluated as

$$P_i = \frac{p_i^\alpha}{\sum_j p_j^\alpha}, \tag{39}$$

where the priority exponent α determines the effect of each transition's priority on the sampling. Another source of bias resulting from stochastic sampling is added by computing *important sampling weights* ω_i, defined as

$$\omega_i = \left(\frac{1}{N\, P_i}\right)^\beta, \tag{40}$$

for all the sampled transitions. These weights are then used in evaluating the (weighted) loss for the critic update L_c. The degree of bias compensation is proportional to the importance sampling exponent β, which is set to an initial value β_0 and reduced with a certain decay rate throughout the training process. For stability reasons, rewards and TD-errors are clipped within $[-1, 1]$.

In the case of dealing with infeasibilities without the use of a safety layer, the PER plays a significant role in expediting the training of the TD3 agent and ensuring the feasibility of the agent's action. At the beginning of training, suppose the RL agent observes a driving state s_t and selects an infeasible action to which is added the random noise, resulting in infeasibility in the powertrain. As a consequence of the infeasibility, the agent receives a penalized reward equal to -1, and the episode terminates. Therefore, the Q-value related to that particular state-action pair will converge to a negative value during training. In a future episode, if the RL agent selects a feasible action resulting in a feasible powertrain operation when it observes the same state s_t, the agent must receive a reward equal to or higher than -1. Hence, it is very likely that the temporal difference (TD)-error associated with the feasible s-a pair is higher than that associated with unfeasible pairs. PER allows to sample this particular transition more frequently than uniform sampling, and the agent will learn faster to avoid infeasible actions and converge for the state s.

Moreover, PER quickly rectifies the RL agent's policy to avoid infeasible control even in the later stage of training. Consider the following situation to understand this concept clearly. During an episode in the later stage of training, suppose the training stops at a particular time step due to the agent's infeasible action selection. Suppose the agent had already overcome infeasibility at that time step in previous episodes. In that case, the reason for its infeasible action selection is a poor estimate of the *Q-value* associated with the infeasible action and the state (excluding the noise added to the exploration). Such a poor estimate results in a high TD-error and consequently increases that transition's probability of being sampled from the replay memory. This way, PER can correct the estimation error more quickly than uniform sampling.

V. Physics Informed Exploration (PIE) strategy

The main novelty of this work is the development of a physics-informed exploration strategy to expedite the learning process of avoiding the actions that lead to infeasibility. Relying on physical information about the infeasible conditions encountered during previous steps of the training phase, the proposed PIE leads the agent towards the exploration of more probably feasible actions. The proposed exploration strategy can be adopted under the realistic assumption that, after encountering an infeasible condition, the cause of infeasibility can be identified. As previously mentioned, infeasible conditions can be caused by several factors related to the operational limits of the HEV components; indeed, the battery, ICE, and electric machines powers need to be compliant with the respective operational limits.

Before describing the workflow of the PIE strategy, some useful quantities are defined. In the PIE framework, the output power P_{out} is discretized with a uniformly spaced grid $\mathcal{P}_{\text{out}}$ of n_P values ranging from $P_{\text{out,min}}$ to $P_{\text{out,max}}$. For each value of the discretized grid, a counter ($\kappa(P_{\text{out}})$) is defined and initialized to zero before the training phase. Moreover, an *expected feasible action set*, which is defined as the set of actions (P_{ICE}) that the PIE strategy can select an action from, is associated with each value in $\mathcal{P}_{\text{out}}$. Each expected feasible action set is bounded by a lower $a_{\text{lb}}(P_{\text{out}})$ and an upper

TABLE III
LIST OF HYPERPARAMETERS USED IN NORMAL AND PRIORITIZED EXPERIENCE REPLAY-ASSISTED TD3 AGENT

Hyperparameters for TD3 agent	Symbol	Value with safety layer	Value without safety layer
Discount factor	γ	0.99	0.99
Standard deviation of exploration noise	σ	0.2	0.05
Clipping standard noise	c	inf	0.2
Target policy noise standard deviation	σ_t	0.1	0.05
Clipping target policy noise	c_t	0.2	0.1
Hidden layer for actor		2	2
Hidden layer for critic		2	2
Learning rate	α	0.0001	0.0001
Tau for target update	τ	0.005	0.005
Replay memory size	$\mathcal{N}$	50000	50000
Learning starts (number of steps)	s_l	1500	1500
Agent action starts (number of steps)	s_a	2000	2000
Mini-batch size	n	32	32
Delay for target and actor update	δ	2	2
Hyperparameters for Prioritized experience replay			
Exploration noise variance	ϵ	-	0.0001
Priority exponent	α	-	0.6
Initial value of the importance sampling exponent	β_0	-	0.4
Decay rate of the importance sampling exponent	$\Delta\beta$	-	0.00001

$a_{\rm ub}(P_{\rm out})$ limit initialized to -1 and +1[2], respectively, since no information about the feasible limits of the action variable is available to the agent at the beginning of the training phase.

In the remainder of this section, the main workflow of the power-based PIE is explained, and the pseudocode of the PIE is shown in Algorithm 2.

At time step t, the requested output power $P_{\rm out}(t)$ is computed and clipped to the closest value in $\mathcal{P}_{\rm out}$. If the counter $\kappa(P_{\rm out}(t))$ is equal to zero, the agent selects an action with added random noise, according to the standard TD3 exploration strategy described in Section IV. If the selected action leads to an infeasible condition, the episode is concluded, and the following PIE parameters are updated: the counter $\kappa(P_{\rm out})$ is increased by 1 and the expected feasible action set bounds are updated based on the cause of infeasibility.

If, on the other hand, the counter $\kappa(P_{\rm out})$ is higher than zero, the action is not directly chosen by the agent, but is randomly sampled between the actual limits of the expected feasible action set. If the action selected in this way leads once again to infeasible conditions, the infeasibility counter and the expected feasible action set limits are updated again. Otherwise, if the selected action is feasible, the infeasibility counter is reset to 0.

If the infeasible condition is due to excessive battery or EMs power requests ($f_{\rm ICE,lb}$, $f_{\rm gen,ub}$, $f_{\rm mot,ub}$, $f_{\rm batt,ub}$), the lower bound $a_{\rm lb}(P_{\rm out})$ of the expected feasible action set is set equal to the infeasible action selected by the agent. Indeed, in this case it can be established that the selected ICE power is too low for the considered value of $P_{\rm out}$; therefore, the agent can be forced to select a higher ICE power in future episodes when faced with the same $P_{\rm out}$. On the other hand, if the infeasibility is due to too low battery or electric motors powers ($f_{\rm ICE,ub}$, $f_{\rm gen,lb}$, $f_{\rm mot,lb}$, $f_{\rm batt,lb}$), the upper limit $a_{\rm ub}(P_{\rm out})$ of the expected feasible action set is set equal to the action that was chosen by the agent. This follows from a similar rationale as for the previous case.

In essence, the safe and feasible action space corresponding to each discrete output power is acquired through the PIE strategy, which involves continuous updates to the lower and upper bounds of the action variable corresponding to every discrete output power ($a_{\rm lb}(P_{\rm out})$ and $a_{\rm ub}(P_{\rm out})$). Unlike conventional safety layers, where the agent is solely informed of the infeasibility of a selected action, our proposed PIE framework provides additional information to the agent, specifically about the powertrain component, which is responsible for constraint violation, and the precise details regarding the nature of the constraint. This distinctive feature contributes to the physics-informed nature of our training process.

Moreover, the PIE and PER perfectly complement each other, enhancing the learning ability of the agent to avoid infeasibilities. Indeed, while PIE leads to faster exploration of feasible actions, the PER helps to quickly update the policy to avoid infeasible actions.

Moreover, it is worth mentioning that the PIE is active only during the training phase and is not adopted in the testing phase. Indeed, during testing, the agent directly selects the action at each time step with pure exploitation, not applying any exploration strategy.

[2]Since the actions are normalized between -1 and 1, $a=-1$ corresponds to $P_{\rm ICE}=P_{\rm ICE,min}$ and $a=-1$ corresponds to $P_{\rm ICE}=P_{\rm ICE,max}$

Algorithm 2 Physics informed exploration strategy (PIE)

Initialize: $\kappa(P_{\text{out}}) = 0, \forall P_{\text{out}}$;
Initialize: $a_{\text{lb}}(P_{\text{out}}) = -1$ and $a_{\text{ub}}(P_{\text{out}}) = 1$, $\forall P_{\text{out}}$;
while $n < N$ **do** ▷ n is the training episode number
  Initialize: $f_{\text{inf}} = 0$ ▷ f_{inf} is the infeasibility flag
  while $t < T \wedge f_{\text{inf}} = 0$ **do**
    Compute $P_{\text{out}}(t)$ and find the correspondent $P_{\text{out}}(t)$
    if $\kappa(P_{\text{out}}(t)) > 0$ **then**
      Execute a random action a between the actual expected feasible action limits $(a_{\text{lb}}(P_{\text{out}}(t)) < a < a_{\text{ub}}(P_{\text{out}}(t)))$
    else
      Obtain action a based on the state s_t through actor network
      Add gaussian noise $\mathcal{N}$ to the action a for exploration
      Execute action plus noise $a + \mathcal{N}$
    end if
    if a is infeasible **then**
      $\kappa(P_{\text{out}}(t)) = \kappa(P_{\text{out}}(t)) + 1$
      if $f_{\text{ICE,lb}}$, $f_{\text{gen,ub}}$, $f_{\text{mot,ub}}$, or $f_{\text{batt,ub}}$ **then**
        Update maximum expected feasible action: $a_{\text{lb}}(P_{\text{out}}(t)) = \mathcal{A}$
      else
        Update minimum expected feasible action $a_{\text{ub}}(P_{\text{out}}(t)) = \mathcal{A}$
      end if
      $f_{\text{inf}} = 1$
    end if
    $n \leftarrow n + 1$
  end while
end while

VI. Results and Discussion

In this section, the performance of the RL-based EMS with the proposed exploration strategy will be compared to the baseline RL-based EMS, i.e., the RL with a safety layer. In this work, a co-simulation framework between Simulink® and Python is developed to explore the performance of different RL-based EMSs. The RL-based EMSs are implemented in Python, while the real-time capable 3rd generation Toyota Prius-alike vehicle model is developed in the Simulink® environment. The bidirectional data exchange between Python and MATLAB/Simulink is obtained using the MATLAB Engine API for Python [54], which provides a Python package for calling MATLAB as a computational engine. A few performance metrics, such as the ability to complete the drive cycle, convergence agility, cumulative reward, charge sustainability, and fuel economy, have been identified for the comparative study.

Four different configurations of the RL agent have been designed for the comparative study:

- agent#1: agent with a safety layer
- agent#2: agent with no safety layer, no PER, no PIE
- agent#3: agent with no safety layer, with PER, but no PIE
- agent#4: agent with no safety layer, with PER and PIE

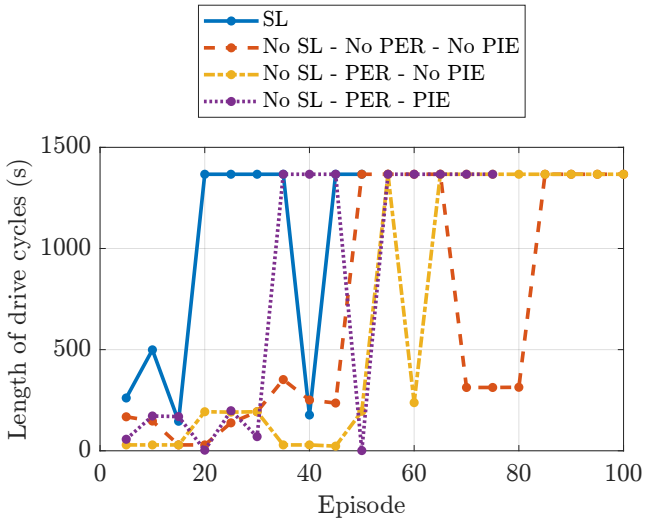

Fig. 8. Comparison of drive cycle's completion capability among four TD3-based RL frameworks when the final objective of the EMS is only to satisfy charge sustainability.

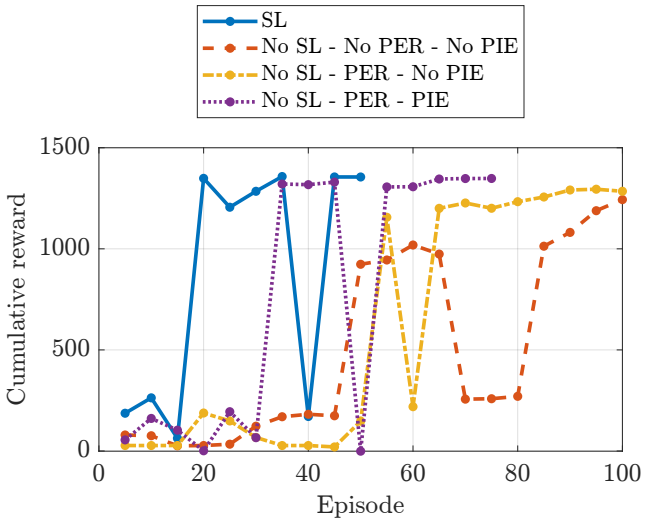

Fig. 9. Comparison of convergence capability among four TD3-based RL frameworks when the final objective of the EMS is only to satisfy charge sustainability.

Figs. 8, 9, 10, and 11 are associated with the first reward function. Figs. 12, 13, and 14 are associated with the second reward function. All the four RL agents are trained over the Urban Dynamometer Driving Schedule (UDDS) drive cycle.

A. SOC-oriented reward

Since the near-optimal policy with the SOC-oriented reward function is easier to learn, the agent training is stopped after 15 hours or when the number of training episodes is 100.

First, the ability to complete the driving cycle is assessed for the 4 agents with the SOC-oriented reward. The use of the safety layer expedites the training completion. With the presence of the safety layer, the environment never encounters an infeasible control action from agent#1. Hence, a particular training episode does not terminate due to the selection of an infeasible control action by agent#1, but is stopped when the battery SOC violates the permissible limits. For the other agents, episode termination can occur due to infeasible action

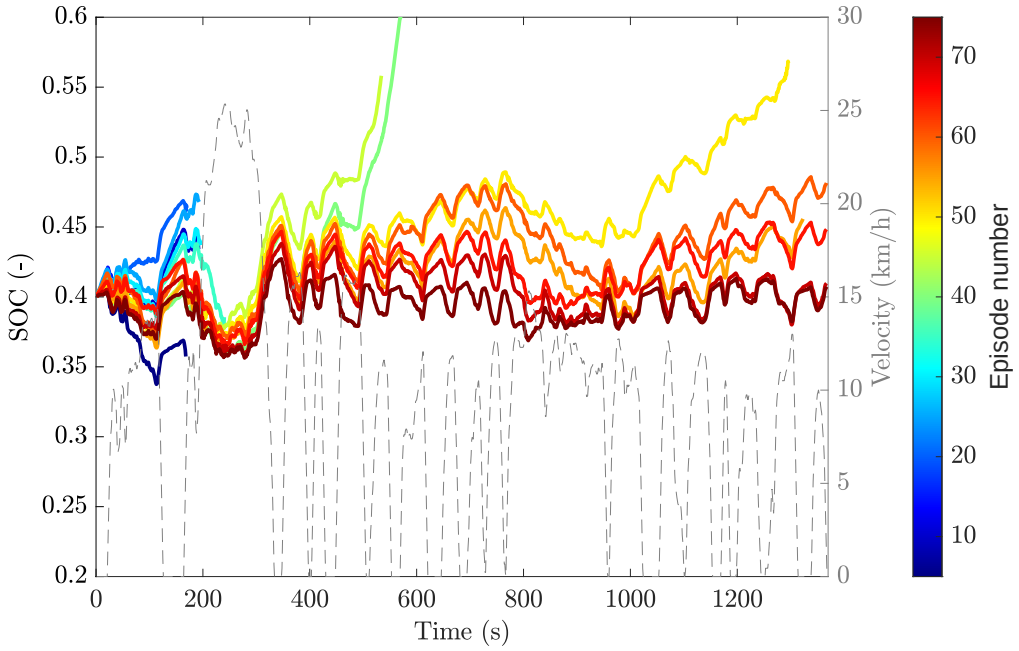

Fig. 10. Training progress of the proposed DRL-based agent (no safety layer, with PER and PIE) in terms of satisfying the charge sustenance and capability of completing the drive cycle. This training is conducted in the co-simulation between Python and Simulink®.

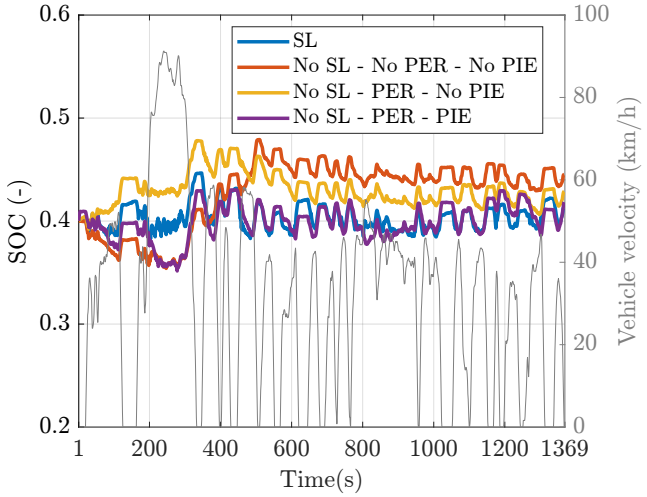

Fig. 11. Comparison of charge sustainability among four TD3-based RL frameworks for UDDS cycle.

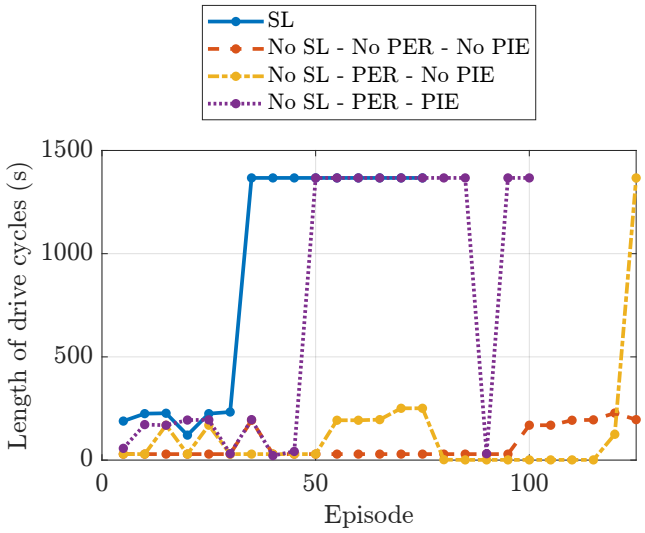

Fig. 12. Comparison of drive cycle's completion capability among four TD3-based RL frameworks when the final objective of the EMS is to satisfy charge sustainability and minimize fuel consumption.

selection and battery SOC limit violation. Therefore, the probability of completing a training episode without termination is highest for agent#1 and then in the chronological order of agent#4, agent#3, and agent#2, as shown in Fig. 8. The use of PER and PIE proves to be helpful to avoid infeasible conditions.

However, the comprehensive comparative study is incomplete without comparing how the cumulative rewards of these four agents converge. Among these four agents, agent#1 achieves convergence the fastest, as shown in Fig. 9, because it has the most interactions with the environment. Indeed, the number of episodes needed to achieve convergence depends on the number of interactions between the agent and the environment. Agents #2, #3, and #4 encounter fewer interactions with the environment than agent #1 during the earlier training episodes due to the early termination of training episodes caused by the selection of infeasible control action. As agents #2, #3, and #4 learn to avoid infeasible control actions over the episodes, the number of interactions with the environment increases and converges to near-optimality.

In terms of performance, the comparison between agents #2 and #3 elucidates the advantage of PER while converging to near-optimality. With PER, the cumulative reward increases quickly, and the cumulative reward of the final episodes achieved by agent#3 is always higher than agent#2. Throughout the training, agent#4 collects more cumulative reward per

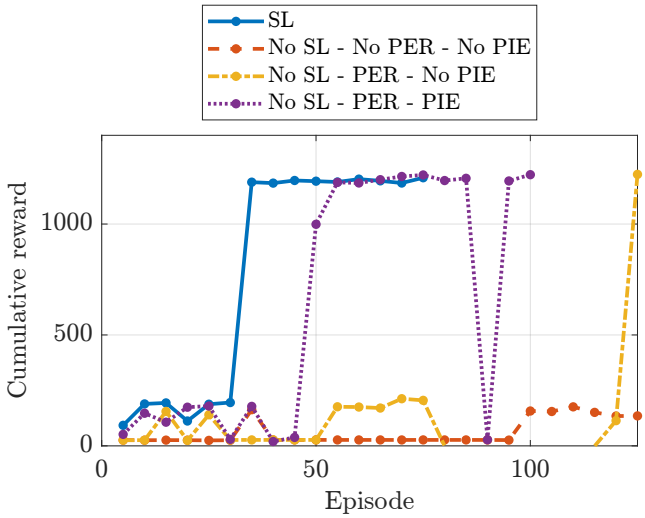

Fig. 13. Comparison of convergence capability among four TD3-based RL frameworks when the final objective of the EMS is to satisfy charge sustainability and minimize fuel consumption.

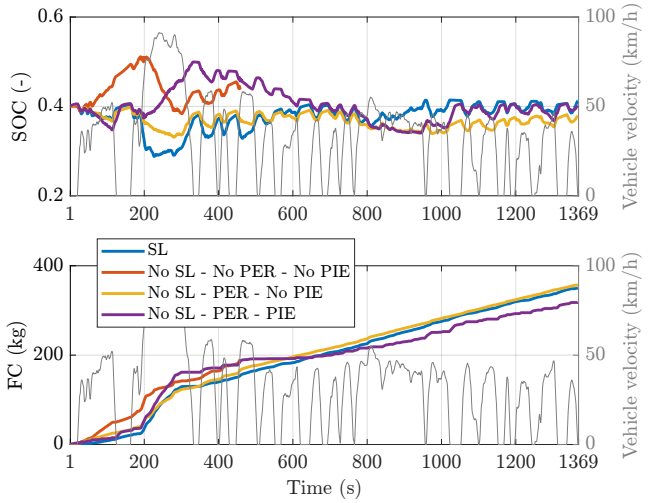

Fig. 14. Comparison of charge sustainability and fuel consumption among four TD3-based RL frameworks for UDDS cycle.

episode than agents #2 and #3, as shown in Fig. 9. Also, the final cumulative reward of agent#4 asymptotically converges to the final cumulative reward of agent#1, which justifies the application of PIE and PER together to replace the safety layer in RL-based agents.

Fig. 10 elaborates on the training progress of agent#4 by showing its evolution towards completing the driving cycle and subsequently achieving charge sustainability at the end. The evolution of the SOC profile during training indicates that agent#4 is learning well. The agent prioritizes avoiding infeasible control actions' selection in the initial training episodes, from 1^{st} to 45^{th} episodes. In contrast, the second half of training, from 46^{th} to 100^{th} episodes, prioritizes both avoiding infeasible control actions' selection and achieving charge sustainability.

Fig. 11 shows the comparison among four RL-based agents studied in this article based on their competence in maintaining charge sustainability after training completion. The claim on the superiority of agents #1 and #4 over #2 and #3 made in Fig. 9 is again corroborated here in Fig. 11. The final value of SOC for the two best frameworks (agents #1 and #4) is much closer to the target value (0.4).

B. FC-oriented reward

The agents are then trained with the second reward function. Fig. 12 compares the four agents from the perspective of the ability to complete the drive cycle with the second reward function. Since the reward function employed in this case is more complex, more time is granted to the agent training. Indeed, training is stopped after 24 hours or when the number of training episodes is 125. Similar to Fig. 8, Fig. 12 depicts the superiority of agent#1 in its ability to complete the drive cycle with the least training episodes.

Agent#2 could not complete the driving cycle even once. This agent is ineffective in solving the more complex reward problem within 125 episodes and would need more episodes before it learns to complete a drive cycle without selecting any infeasible control action. Agent#3 also struggles to handle the complex reward function during most of the training phase and manages to complete the drive cycle without selecting any infeasible control action only in the 125^{th} episode. The real advantage of the PIE is revealed in this figure since agent#4 can complete the drive cycle in all the training episodes after 50^{th}. The only exception is episode 90 where an infeasibility condition is encountered before reaching the end of the driving cycle. However, thanks to the fast learning abilities due to PER, the agent is capable of completing the cycle in the subsequent episodes.

Fig. 13 furnishes the convergence performance of the agents under the new reward function. As usual, agent#1 converges the fastest (within 30 episodes) to near-optimality. Agent#2 struggles the most to converge to near-optimality and settles for sub-optimal performance within 125 training episodes. Agent#3 reaches near-optimal performance only in the final training episode. Despite its delayed convergence, agent#4 converges to a cumulative reward value slightly higher than agent#1. Agent#4 outperforming agent#1 is a significant result and can be explained by theoretical comments from [24].

The fuel consumption and charge sustainability performances of agents #4 and #1 are compared to verify the superiority of agent#4 over agent#1. Both agents performed remarkably well in achieving charge sustainability, as shown in the first subplot of Fig. 14. However, agent#4 outperforms agent#1 in fuel consumption performance by consuming 33 grams less fuel for urban dynamometer driving schedule (UDDS), as shown in the second subplot of Fig. 14 and in Table IV.

Since the FC-oriented reward function leads to a more complex policy, the performance of the trained RL agents needs to be assessed over a driving cycle different from the training one, to test the adaptability of the RL agents to unknown driving conditions. Therefore, the 4 agents are trained on UDDS and then tested over the Artemis Urban Driving Cycle (AUDC) which is representative of an urban driving scenario. Once again, agent#4 outperforms all the other RL agents, achieving charge sustainability and lower FC (as

TABLE IV
FUEL CONSUMPTION COMPARISON AMONG THE 4 RL AGENTS WITH FC-ORIENTED REWARD

Driving cycle	Agent#1 (with a safety layer)	Agent#2 (with no safety layer, no PER, and no PIE)	Agent#3 (with no safety layer, with PER, but no PIE)	Agent#4 (with no safety layer, with PER, and PIE)
UDDS	350.7 grams	-	357.5 grams (+1.9%)	318.3 grams (-9.2%)
AUDC	212.9 grams	-	207.1 grams (-2.7%)	167.9 grams (-21.1%)

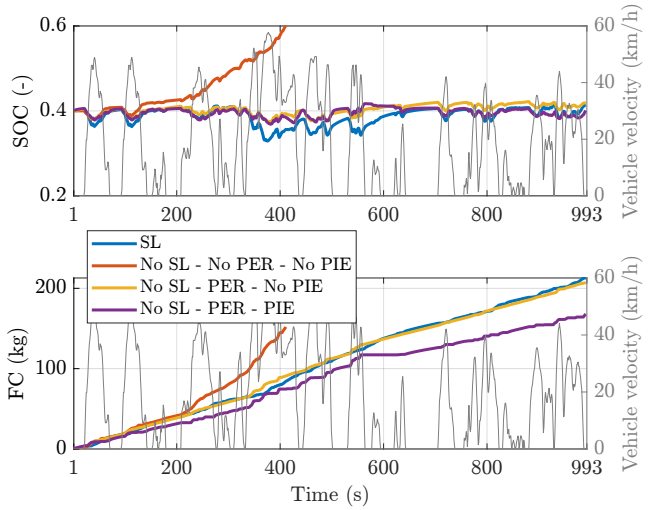

Fig. 15. Comparison of charge sustainability and fuel consumption among four TD3-based RL frameworks for AUDC cycle.

shown in Tab.IV).

Finally the real-time capability of the proposed algorithm are assessed. All simulations were run on a computer with an 11th Gen Intel(R) Core(TM) i7-1165G7 processor running at 2.8 GHz, with 64GB of RAM. The average computational time for one successfully concluded training episode over the UDDS is approximately equal to 700 seconds. Since the computational time required to complete a whole simulation of the UDDS cycle is shorter than the real duration of the UDDS cycle, i.e., 1369 seconds, the possibility to run the algorithm in real-time is proven.

VII. CONCLUSION

The article proposes a novel physics-informed exploration technique for RL controllers/agents solving practical and constrained optimization problems. The numerical experiments has demonstrated that the proposed exploration technique for an RL agent can be an innovative replacement for safe learning, which is otherwise mandatory for constrained MDPs. A TD3-based framework with a safety layer is designed as a baseline energy management controller for a power-split HEV similar to the third-generation Toyota Prius. The main objective of this work is to prove that a near-optimal control of a practical and constrained MDP, like an energy management problem of an HEV, can be solved without a safety layer. To this end, the authors designed a novel TD3-based framework without any safety layer which instead combines prioritized experience replay and physics-informed exploration. A state-of-the-art platform co-simulating between Python and Simulink® is employed to simulate the baseline and proposed TD3-based energy management framework. Simulation results show although the proposed framework has a delayed convergence compared to the baseline, it outperforms the baseline framework with safety layer by 10.1% in terms of near-optimality of fuel consumption performance. The proposed framework converges within a lower number of training episodes.

REFERENCES

[1] A. Biswas and A. Emadi, "Energy management systems for electrified powertrains: State-of-the-art review and future trends," *IEEE Trans. Veh. Technol.*, vol. 68, no. 7, pp. 6453–6467, July 2019.

[2] F. Miretti and D. Misul, "Robust modeling for optimal control of parallel hybrids with dynamic programming," in *2022 IEEE Transportation Electrification Conference & Expo (ITEC)*, June 2022, pp. 1015–1020.

[3] W. Hong, I. Chakraborty, H. Wang, and G. Tao, "Co-optimization scheme for the powertrain and exhaust emission control system of hybrid electric vehicles using future speed prediction," *IEEE Transactions on Intelligent Vehicles*, vol. 6, no. 3, pp. 533–545, 2021.

[4] A. Biswas, P. G. Anselma, A. Rathore, and A. Emadi, "Effect of coordinated control on real-time optimal mode selection for multi-mode hybrid electric powertrain," *Appl. Energy*, vol. 289, p. 116695, 2021.

[5] M. Acquarone, A. Borneo, and D. A. Misul, "Acceleration control strategy for battery electric vehicle based on deep reinforcement learning in v2v driving," in *2022 IEEE Transportation Electrification Conference & Expo (ITEC)*, June 2022, pp. 202–207.

[6] P. G. Anselma, P. Kollmeyer, J. Lempert, Z. Zhao, G. Belingardi, and A. Emadi, "Battery state-of-health sensitive energy management of hybrid electric vehicles: Lifetime prediction and ageing experimental validation," *Appl. Energy*, vol. 285, p. 116440, 2021.

[7] A. M. Ali and B. Moulik, "On the role of intelligent power management strategies for electrified vehicles: A review of predictive and cognitive methods," *IEEE Transactions on Transportation Electrification*, vol. 8, no. 1, pp. 368–383, 2022.

[8] H. He, X. Meng, Y. Wang, A. Khajepour, X. An, R. Wang, and F. Sun, "Deep reinforcement learning based energy management strategies for electrified vehicles: Recent advances and perspectives," *Renewable and Sustainable Energy Reviews*, vol. 192, p. 114248, 2024.

[9] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, Jan 2016.

[10] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[11] P. R. Wurman, S. Barrett, K. Kawamoto, J. MacGlashan, K. Subramanian, T. J. Walsh, R. Capobianco, A. Devlic, F. Eckert, F. Fuchs, L. Gilpin, P. Khandelwal, V. Kompella, H. Lin, P. MacAlpine, D. Oller, T. Seno, C. Sherstan, M. D. Thomure, H. Aghabozorgi, L. Barrett, R. Douglas, D. Whitehead, P. Dürr, P. Stone, M. Spranger, and H. Kitano, "Outracing champion gran turismo drivers with deep reinforcement learning," *Nature*, vol. 602, no. 7896, pp. 223–228, Feb 2022.

[12] R. C. Hsu, C.-T. Liu, and D.-Y. Chan, "A reinforcement-learning-based assisted power management with qor provisioning for human–electric hybrid bicycle," *IEEE Transactions on Industrial Electronics*, vol. 59, no. 8, pp. 3350–3359, Aug 2012.

[13] R. Johri and Z. Filipi, "Self-learning neural controller for hybrid power management using neuro-dynamic programming," in *10th International Conference on Engines Vehicles*. SAE International, sep 2011. [Online]. Available: https://doi.org/10.4271/2011-24-0081

[14] G. Du, Y. Zou, X. Zhang, L. Guo, and N. Guo, "Heuristic energy management strategy of hybrid electric vehicle based on deep reinforcement learning with accelerated gradient optimization," *IEEE Transactions on Transportation Electrification*, vol. 7, no. 4, pp. 2194–2208, 2021.

[15] A. Biswas, P. G. Anselma, and A. Emadi, "Real-time optimal energy management of multi-mode hybrid electric powertrain with online trainable asynchronous advantage actor-critic algorithm," *IEEE Transactions on Transportation Electrification*, pp. 1–1, 2021.

[16] R. Lian, H. Tan, J. Peng, Q. Li, and Y. Wu, "Cross-type transfer for deep reinforcement learning based hybrid electric vehicle energy management," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 8, pp. 8367–8380, Aug 2020.

[17] H. He, Y. Wang, J. Li, J. Dou, R. Lian, and Y. Li, "An improved energy management strategy for hybrid electric vehicles integrating multistates of vehicle-traffic information," *IEEE Transactions on Transportation Electrification*, vol. 7, no. 3, pp. 1161–1172, Sep. 2021.

[18] A. Mekrache, A. Bradai, E. Moulay, and S. Dawaliby, "Deep reinforcement learning techniques for vehicular networks: Recent advances and future trends towards 6g," *Vehicular Communications*, vol. 33, p. 100398, 2022.

[19] P. Dong, J. Zhao, X. Liu, J. Wu, X. Xu, Y. Liu, S. Wang, and W. Guo, "Practical application of energy management strategy for hybrid electric vehicles based on intelligent and connected technologies: Development stages, challenges, and future trends," *Renewable and Sustainable Energy Reviews*, vol. 170, p. 112947, 2022.

[20] X. Tang, J. Chen, K. Yang, M. Toyoda, T. Liu, and X. Hu, "Visual detection and deep reinforcement learning-based car following and energy management for hybrid electric vehicles," *IEEE Transactions on Transportation Electrification*, vol. 8, no. 2, pp. 2501–2515, June 2022.

[21] H. Hu, W.-W. Yuan, M. Su, and K. Ou, "Optimizing fuel economy and durability of hybrid fuel cell electric vehicles using deep reinforcement learning-based energy management systems," *Energy Conversion and Management*, vol. 291, p. 117288, 2023.

[22] B. Hu and J. Li, "An adaptive hierarchical energy management strategy for hybrid electric vehicles combining heuristic domain knowledge and data-driven deep reinforcement learning," *IEEE Transactions on Transportation Electrification*, vol. 8, no. 3, pp. 3275–3288, Sep. 2022.

[23] H. Zhang, B. Chen, N. Lei, B. Li, R. Li, and Z. Wang, "Integrated thermal and energy management of connected hybrid electric vehicles using deep reinforcement learning," *IEEE Transactions on Transportation Electrification*, pp. 1–1, 2023.

[24] A. Biswas, Y. Wang, and A. Emadi, "Effect of immediate reward function on the performance of reinforcement learning-based energy management system," in *2022 IEEE Transportation Electrification Conference & Expo (ITEC)*, June 2022, pp. 1021–1026.

[25] C. Qi, C. Song, D. Wang, F. Xiao, L. Jin, and S. Song, "Action advising and energy management strategy optimization of hybrid electric vehicle agent based on uncertainty analysis," *IEEE Transactions on Transportation Electrification*, pp. 1–1, 2023.

[26] L. Brunke, M. Greeff, A. W. Hall, Z. Yuan, S. Zhou, J. Panerati, and A. P. Schoellig, "Safe learning in robotics: From learning-based control to safe reinforcement learning," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 5, no. 1, pp. 411–444, 2022.

[27] A. Krajna, M. Brcic, T. Lipic, and J. Doncevic, "Explainability in reinforcement learning: perspective and position," 2022.

[28] D. P. Bertsekas, *Constrained optimization and Lagrange multiplier methods*, ser. Computer Science and Applied Mathematics. Boston, MA, USA: Academic press, 1982.

[29] S. Zhang, R. Jia, H. Pan, and Y. Cao, "A safe reinforcement learning-based charging strategy for electric vehicles in residential microgrid," *Applied Energy*, vol. 348, p. 121490, 2023.

[30] H. Li, Z. Wan, and H. He, "Constrained ev charging scheduling based on safe deep reinforcement learning," *IEEE Transactions on Smart Grid*, vol. 11, no. 3, pp. 2427–2439, May 2020.

[31] G. Chen and X. Shi, "A deep reinforcement learning-based charging scheduling approach with augmented lagrangian for electric vehicle," 2022.

[32] H. Zhang, J. Peng, H. Tan, H. Dong, and F. Ding, "A

deep reinforcement learning-based energy management framework with lagrangian relaxation for plug-in hybrid electric vehicle," *IEEE Transactions on Transportation Electrification*, vol. 7, no. 3, pp. 1146–1160, Sep. 2021.

[33] Z. Yan and Y. Xu, "Real-time optimal power flow: A lagrangian based deep reinforcement learning approach," *IEEE Transactions on Power Systems*, vol. 35, no. 4, pp. 3270–3273, 2020.

[34] A. Wachi, Y. Sui, Y. Yue, and M. Ono, "Safe exploration and optimization of constrained mdps using gaussian processes," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, Apr. 2018. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/12103

[35] K. Srinivasan, B. Eysenbach, S. Ha, J. Tan, and C. Finn, "Learning to be safe: Deep RL with a safety critic," *CoRR*, vol. abs/2010.14603, 2020.

[36] A. Kumar, A. Zhou, G. Tucker, and S. Levine, "Conservative q-learning for offline reinforcement learning," *CoRR*, vol. abs/2006.04779, 2020.

[37] H. Bharadhwaj, A. Kumar, N. Rhinehart, S. Levine, F. Shkurti, and A. Garg, "Conservative safety critics for exploration," *CoRR*, vol. abs/2010.14497, 2020.

[38] H. Krasowski, J. Thumm, M. Müller, L. Schäfer, X. Wang, and M. Althoff, "Provably safe reinforcement learning: Conceptual analysis, survey, and benchmarking," *Transactions on Machine Learning Research*, 2023, survey Certification.

[39] Z. E. Liu, Q. Zhou, Y. Li, S. Shuai, and H. Xu, "Safe deep reinforcement learning-based constrained optimal control scheme for hev energy management," *IEEE Transactions on Transportation Electrification*, pp. 1–1, 2023.

[40] B. Hu and J. Li, "An adaptive hierarchical energy management strategy for hybrid electric vehicles combining heuristic domain knowledge and data-driven deep reinforcement learning," *IEEE Transactions on Transportation Electrification*, vol. 8, no. 3, pp. 3275–3288, Sep. 2022.

[41] J. Wu, C. Huang, H. He, and H. Huang, "Confidence-aware reinforcement learning for energy management of electrified vehicles," *Renewable and Sustainable Energy Reviews*, vol. 191, p. 114154, 2024.

[42] G. Dulac-Arnold, N. Levine, D. J. Mankowitz, J. Li, C. Paduraru, S. Gowal, and T. Hester, "Challenges of real-world reinforcement learning: definitions, benchmarks and analysis," *Machine Learning*, vol. 110, no. 9, pp. 2419–2468, Sep 2021.

[43] A. Biswas, A. Rathore, and A. Emadi, "Coordinated clutch actuation for drivability improvement and energy management of a novel multi-mode hybrid electric vehicle and hil validation," *Energy Conversion and Management*, vol. 287, p. 117060, 2023.

[44] A. Biswas, O. Rane, A. Rathore, P. G. Anselma, Y. Wang, J. Toller, J. Roeleveld, B. Wasacz, and A. Emadi, "Energy management system for input-split hybrid electric vehicle (si-evt) with dynamic coordinated control and mode-transition loss," in *WCX SAE World Congress Experience*. SAE International, mar 2022. [Online]. Available: https://doi.org/10.4271/2022-01-0674

[45] M. Olszewski, "Evaluation of the 2010 Toyota Prius Hybrid Synergy Drive System," Mar 2011.

[46] P. Ahmadizadeh, B. Mashadi, and D. Lodaya, "Energy management of a dual-mode power-split powertrain based on the pontryagin's minimum principle," *IET Intelligent Transport Systems*, vol. 11, no. 9, pp. 561–571, 2017.

[47] T. Gray and M. Shirk, "2010 Toyota Prius VIN 0462 hybrid electric vehicle battery test results," Idaho National Lab.(INL), Idaho Falls, ID (United States), Tech. Rep., 2013.

[48] S. Onori, L. Serrao, and G. Rizzoni, "Hybrid electric vehicles: Energy management strategies," 2016.

[49] G. E. Monahan, "State of the art—a survey of partially observable markov decision processes: theory, models, and algorithms," *Management science*, vol. 28, no. 1, pp. 1–16, 1982.

[50] M. Acquarone, C. Maino, D. Misul, E. Spessa, A. Mastropietro, L. Sorrentino, and E. Busto, "Influence of the reward function on the selection of reinforcement learning agents for hybrid electric vehicles real-time control," *Energies*, vol. 16, no. 6, p. 2749, 2023.

[51] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proceedings of the 35th International Conference on Machine Learning*, vol. 80. PMLR, 10–15 Jul 2018, pp. 1587–1596.

[52] T. Joshi, S. Makker, H. Kodamana, and H. Kandath, "Twin actor twin delayed deep deterministic policy gradient (tatd3) learning for batch process control," *Computers & Chemical Engineering*, vol. 155, p. 107527, 2021.

[53] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," 2015. [Online]. Available: https://arxiv.org/abs/1511.05952

[54] MATLAB, "Calling matlab from python," https://it.mathworks.com/help/matlab/matlab-engine-for-python.html, 2023.