IEEE Open Journal of
Power Electronics

Received 18 January 2026; revised 12 March 2026; accepted 9 April 2026. Date of publication 16 April 2026;
date of current version 30 April 2026. The review of this article was arranged by Associate Editor Zeng Liu.

Digital Object Identifier 10.1109/OJPEL.2026.3684829

Model-Free Deep Reinforcement Learning Control for Grid-Connected Packed U-Cell Multilevel Inverters

ALAMERA NOURAN ALQUENNAH [1,1] (Graduate Student Member, IEEE), TASSNEEM ZAMZAM [1,2] (Graduate Student Member, IEEE), AHMED KOUZOU [1,2] (Graduate Student Member, IEEE), AZADEH KERMANSARAVI [3] (Member, IEEE), MOHAMED TRABELSI [4] (Senior Member, IEEE), SERTAC BAYHAN [5] (Senior Member, IEEE), HAITHAM ABU-RUB [5,6] (Fellow, IEEE), ALI GHRAYEB [6] (Fellow, IEEE), HANI VAHEDI [3,7] (Senior Member, IEEE), AND SUNIL KHATRI VAHEDI [1] (Senior Member, IEEE)

[1]Electrical and Computer Engineering Department, Texas A&M University, College Station, TX 77843-3128 USA
[2]Electrical and Computer Engineering Department, Texas A&M University at Qatar, Doha 23874, Qatar
[3]Delft University of Technology, 2628CD Delft, The Netherlands
[4]Kuwait College of Science and Technology, Kuwait City 81002, Kuwait
[5]Qatar Environmental and Energy Research Institute, Hamad Bin Khalifa University, Doha 34110, Qatar
[6]College of Science and Engineering, Hamad Bin Khalifa University, Doha 34110, Qatar
[7]College of Engineering and Energy, Abdullah Al Salem University, Khaldiya 72303, Kuwait

CORRESPONDING AUTHOR: ALAMERA NOURAN ALQUENNAH (e-mail: alamera_nq@tamu.edu)

This wok was supported in part by Open Access through Qatar National Library (QNL), in part by Qatar Research, Development and Innovation (QRDI) Council under Grant ARG01-0428-230023, and in part by The analysis of the simulation part (Section IV) under Project CN23-13EE1882 funded by Kuwait Foundation for the Advancement of Sciences (KFAS).

ABSTRACT This paper proposes an innovative model-free deep reinforcement learning-based controller (RL-C) for a grid-connected 5-level packed-U-cell (PUC5) multilevel inverter (MLI). The controller is designed to deliver a high-quality grid current while maintaining the PUC5 floating capacitor voltage at its reference level. In addition, the proposed controller supports both active and reactive power exchanges, adapts to variations in voltage and current references, and remains robust under grid voltage variations. The RL agent learns optimal switching actions through direct interaction with the PUC5 system, eliminating the need for data collection or reliance on existing control models. An Actor-Critic architecture is adopted, and the Proximal Policy Optimization (PPO) algorithm is applied for training (offline) using MATLAB/Simulink, where the RL-C is evaluated under diverse PUC5 configurations and operating conditions in the testing phase. The trained agent has been implemented on an Opal-RT real-time system and validated experimentally using a laboratory-made PUC5 prototype. The performance of the proposed RL-C approach is compared to both traditional approaches including finite control set model predictive control, sliding mode control, and PI control, and other state-of-the-art RL algorithms, demonstrating superior generalization and training efficiency. Moreover, a sensitivity analysis quantifying the impact of reward design, state space, network size, and key hyperparameters on convergence and performance is carried out.

INDEX TERMS Packed-U-Cell Inverter, Reinforcement Learning, AI Controllers.

I. INTRODUCTION

Many control strategies have been proposed for multilevel inverters (MLIs) to address the limitations of conventional control structures and to satisfy stringent grid-code requirements under wide operating conditions [1], [2]. In addition to traditional approaches such as Proportional-Integral (PI) control [3], [4] and Sliding Mode Control (SMC) [5], [6], predictive control method, particularly finite control set model predictive control (FCS-MPC), have attracted significant interest due to their ability to incorporate multiple objectives

in a single design framework [7]. In FCS-MPC, the converter model is used to predict the system behavior for all admissible switching configurations, and the optimal switching state is selected by minimizing a predefined cost function [8]. Related formulations are often described under the broader class of model-based predictive control, which is frequently highlighted for its straightforward design and the direct inclusion of multiple control objectives [1].

Recently, artificial intelligence (AI)-based controllers have captured the attention of researchers in the field of power electronics due to their ability to map observed measurements from the system to appropriate control actions. Their robust performance in both steady-state and dynamic conditions, along with their adaptability to changes in operating conditions, further underscores their utility. This includes handling grid disturbances and variations in passive components, such as capacitors and inductors.

AI algorithms have been employed in grid-connected MLIs to enhance functionality and performance [2], [9]. The primary objectives are: 1) to directly control the inverter's switches to regulate the capacitors' voltages around their desired reference values and deliver a high quality current to the grid with reduced total harmonic distortion (THD) [10], [11], [12], [13], [14], [15], [16], [17]; 2) to enhance the performance of other control algorithms, such as FCS-MPC and backstepping control, by auto-tuning their parameters to achieve the optimal action for the observed system measurements [18], [19], [20], [21], [22], [23], [24].

Reinforcement learning (RL) [25] is one of the AI methods recently applied for MLIs, including 5-level and 7-level packed U-cell (PUC5 and PUC7, respectively) MLIs [10], [11], [12], [14], [15], 9-level packed-E-cell [13], [20], and 3-level neutral point clamped [14] configurations. An RL-based controller (RL-C) learns to map the optimal control action to the observed system measurements—referred to as RL states—within the grid-connected MLI environment. This learning occurs through iterative interactions between the RL-C, acting as the RL agent, and its environment. During the learning process, the RL agent seeks to maximize a predefined reward function, which guides it towards the desired performance. At each time step, or iteration, the RL agent selects the appropriate action based on the observed states. The efficacy of this action is then evaluated via the reward function, which indicates whether the action meets the control objectives [26]. Designing an RL-C involves several critical steps: defining the learning environment to include the system and any potential operational randomness, identifying the observable states to be provided to the RL-C, establishing the action space from which the RL-C will choose appropriate actions, and creating a reward function. The design of the reward function is particularly crucial as it significantly influences the learning process and the rate of convergence.

Unlike traditional controllers such as PI and SMC, which are commonly tuned around a limited set of operating points, RL-based controllers can optimize performance over variable operating conditions by incorporating operating variability directly into the learning process; this adaptability is further demonstrated in this paper. Compared to other AI-based controllers, such as supervised neural network (NN)-based controllers, the RL-C does not rely on collecting training data based on the performance of other controllers [27], [28], [29], [30]. Therefore, its learning process is independent of the performance limitations inherent in the training data of dependent controllers. Additionally, randomness or reference variations in operating conditions can be directly modeled within the designed environment, eliminating the need for a separate data collection stage for training. In contrast to model-based controllers like FCS-MPC, the RL-C does not require knowledge of the system's model, as it solely relies on mapping the control action to the observed states. Moreover, the FCS-MPC algorithm depends on an approximate discretized version of the system's mathematical model, where its accuracy is influenced by the order of approximation. This is not an issue for small sampling times, but it affects model accuracy at larger time steps [31]. In addition, mismatches between the sizes of passive elements in the physical system and their substituted values in the mathematical model may impact the performance of FCS-MPC. In such cases, RL was applied to auto-tune FCS-MPC and to enhance its robustness against component mismatches [18], [21]. However, previous work has demonstrated the robustness of RL-C in handling variations in passive elements within the model [10], [13], highlighting its generalization capability. In addition, compared to FCS-MPC and Lyapunov-based MPC techniques, RL-C employs the learned agent policy to determine the proper control action without solving an online optimization problem. This advantage becomes more significant as the number of switching configurations increases, as it helps reduce the online computational burden.

On the other hand, this promising model-free algorithm requires a high number of training episodes, which increases with the level of randomness incorporated into the environment. Also, the fed observations to RL-C depend on its actions may lead to temporal correlations in the training process [32]. In addition, the RL-C design involves selecting RL model from a wide array of RL architectures and training algorithms, each potentially suitable depending on the application requirements and system characteristics, including whether the state space and action space are continuous or discrete. The selection of these architectures and the tuning of numerous hyperparameters often rely on a combination of trial and error and empirical insights, underscoring the lack of fixed guidelines in these areas. Moreover, careful design of the training environment is crucial to ensure generalization and prevent overfitting, where the agent learns only a narrow set of operating conditions. Similarly, the design of the reward function significantly influences both the learning process and training convergence. The effectiveness of this function heavily depends on the specific system under study, further highlighting the bespoke nature of RL-C system development.

Examples of recent review studies on RL-based controllers for power electronics include [33], [34]. These surveys cover a

broad range of RL applications, including RL-assisted implementations (e.g., MPC- and SMC-based controllers enhanced by RL) as well as fully RL-based control schemes, which is the category to which the proposed model belongs. The survey in [34] indicates that 64% of the reviewed studies were conducted in simulation only, while 14% were validated using hardware-in-the-loop (HIL), with relatively few studies providing full experimental validation. Only a limited number of works have addressed MLIs, which typically involve larger action and state spaces and multi-objective control requirements. Also, previous RL-C studies did not evaluate the reliability of the designed controllers under grid disturbances, including voltage sags and harmonic distortion [33]. Moreover, prior studies often lack systematic benchmarking of RL-C performance against conventional controllers, as well as comparative evaluation of different RL algorithms. Furthermore, there is no standardized environment or in-depth analysis quantifying the impact of main RL design components, such as reward function formulation and state/action space selection, on closed-loop performance and training convergence. In addition, previous studies did not examine the training cost and online computational burden of different RL algorithms in power electronics, nor did they compare these aspects with traditional control approaches. These aspects are explicitly considered in this study through a detailed sensitivity analysis, providing practical guidance and a useful starting point for future RL-C designs in MLI applications.

The RL-C previously designed for MLIs employed four learning algorithms: Double Deep Q Neural Network (DDQNN) [11], Proximal Policy Optimization (PPO) [10], [12], [13], Deep Deterministic Policy Gradient (DDPG) [17], and Twin Delayed Deep Deterministic Policy Gradient (TD3) [14]. PPO is recognized for its robustness in learning compared to DDQNN, DDPG, and TD3, which often suffer from instability in continuous control applications and are highly sensitive to hyperparameter tuning. Additionally, applying TD3 and DDPG, which are designed for continuous action spaces, to MLI control typically requires an additional modulation stage to generate the actual switching signals. This contrasts with DDQNN and PPO formulations with discrete actions, which can directly map the observed state to a discrete switching state applied to the power electronic converter. With a constant switching-frequency modulation stage, the inverter may switch more frequently than necessary during certain portions of the fundamental cycle, even when additional switching is not required to meet the control objectives. However, the PPO-learned policy does not rely on a modulation stage and can inherently produce a variable switching frequency by selecting switching actions directly from the observed state. Moreover, the PPO and DDQN formulations considered in this work naturally align with digital control in power electronics, where observations are continuous while the action space is discrete (switching states). This formulation is analogous to FCS-MPC, which has demonstrated strong performance in power electronic applications.

The RL-C developed for the PUC5 in [10] was trained and tested for a system integrated with a photovoltaic (PV) source, where the training environment included variations in the current reference but only slight variations in the capacitor voltage reference. Moreover, that design utilized a relatively large neural network [10], [12]. This is in addition to not reporting the RL-C performance with respect to other control algorithms, nor comparing the achieved performance of different RL algorithms under the same problem formulation. Furthermore, the previous work did not analyze the impact of the NN size, the state-space design, the hyperparameter settings, or the reward function design on the RL-C performance and the training process. Instead, the parameters were selected through trial and error or by adopting values reported for other systems. These gaps in previous research have been highlighted in [33] and [34], which emphasize the limited availability of systematic sensitivity studies evaluating how RL design parameters affect the controller performance and convergence.

Therefore, to address the shortcomings of traditional control methods and previous RL-based controllers, and to provide a detailed and reproducible study that can guide RL-based control development for PUC5 and other MLI topologies. The main contributions of this work are summarized as follows:

1) Proposes a model-free RL-C for a single-phase, grid-connected PUC5 MLI using a shallow actor-critic neural-network architecture, which is trained under wide operating conditions, including variations in injected power levels and DC voltage supply.
2) Demonstrates strong robustness and generalization by evaluating the trained RL-C on multiple PUC5 configurations (different L and C values) and operating scenarios that were not included during training.
3) Compared to prior work [10], [11], [12], it reduces both the observation set and the neural-network size, resulting in fewer learnable parameters and a lightweight structure that supports real-time implementation.
4) Compares the proposed RL-C with four established controllers for the PUC system, including FCS-MPC, PI control, and two SMC variants [3], [5], [6], [35].
5) Benchmarks the proposed PPO-based RL-C against three additional RL algorithms under the same problem formulation, and reports both control performance and training-related metrics.
6) Presents a sensitivity analysis quantifying the impact of reward-function design, state-space configuration, neural-network size, and key hyperparameters on training convergence and closed-loop performance.
7) Finally, the proposed RL-C is experimentally validated to confirm its real-time applicability and performance.

The remainder of this paper is organized as follows: Section II introduces the PUC5 system and outlines the control objectives. Section III presents the design of the RL-C along with the training algorithm. Section IV presents simulation results, comparisons with other control strategies, and

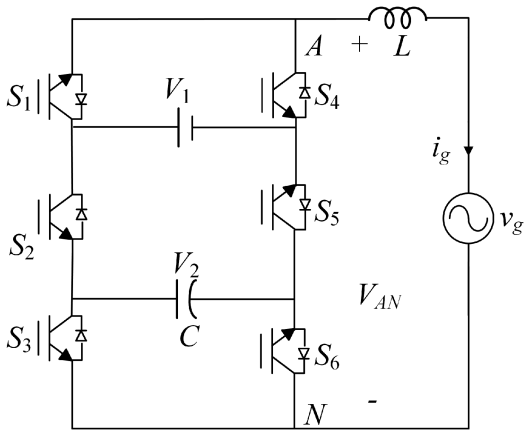

FIGURE 1. The system under study: single phase 5-level packed U cell - Multilevel inverter connected to the grid.

TABLE I PUC Switching Configurations and Their Corresponding Output Voltage

S_1,S_2,S_3	$V_{AN}(V)$	Action	S_1,S_2,S_3	$V_{AN}(V)$	Action
1,0,0	V_1	a_1	1,1,1	0	a_5
1,0,1	$V_1 - V_2$	a_2	0,0,1	$-V_2$	a_6
1,1,0	V_2	a_3	0,1,0	$-V_1 + V_2$	a_7
0,0,0	0	a_4	0,1,1	$-V_1$	a_8

a sensitivity analysis on reward design, state space, network size, and hyperparameters. Section V presents the experimental validation of the proposed RL-C. Finally, Section VI provides a summary and concludes the paper.

II. PUC5 SYSTEM AND THE CONTROL OBJECTIVES

A. PUC5 SYSTEM

The PUC5 system consists of a single-phase PUC5 MLI connected to the grid, with voltage v_g, through a filtering inductor (L), as shown in Fig. 1. The topology of the PUC5 includes two U cells, each equipped with two DC links that can be a DC source or capacitor, depending on the application. In the grid-connected PUC5 application, the first DC link is a DC source with voltage V_1, while the second DC link is an auxiliary capacitor (C) with voltage V_2. The two DC links are connected via two power switches (S_2, S_5), in addition to other power switches that connect them to the positive output terminal of the inverter (S_1, S_4) and two switches that connect them to the negative output terminal (S_3, S_6), respectively.

At each time step (T_s), the switch state configuration (s_i, i = 1 to 6) can be ON ($s_i = 1$) or OFF ($s_i = 0$), under the constraint that no DC link is short-circuited. Consequently, each pair of switches within a cell is controlled in a complementary manner. The valid switching states are listed in Table 1, resulting in eight possible switching configurations. The series connection of V_1 and V_2 at T_s determines the inverter's output voltage, denoted as V_{AN}, and i_g is the current signal fed to the grid. The design requirements, variations, and industrial applications of the PUC topology were thoroughly discussed in [36].

B. CONTROL OBJECTIVES

The first control objective is to generate five DC levels at V_{AN}, achieved by regulating V_2 to $V_2^* = V_1/2$, which defines the desired V_2 value. This regulation results in redundant switching states, which enhance the stability of the system. Accordingly, let e_v be defined as the deviation between V_2 and V_2^* as follows:

$$e_v = |V_2 - V_2^*|. \tag{1}$$

Based on that, the first control objective can be achieved by minimizing e_v.

The second control objective is to feed the grid with a pure sinusoidal signal while minimizing THD. To achieve this, i_g must be controlled by tracking the current reference signal i_g^*. Consequently, e_i, the current error, can be defined as:

$$e_i = |i_g - i_g^*|, \tag{2}$$

where i_g^* is given by:

$$i_g^* = I_p^* \times \sin(\theta_g + \phi). \tag{3}$$

In this equation, I_p^* represents the peak of the current reference, θ_g is the phase of v_g, and ϕ is a phase shift value. The value of I_p^* is determined based on grid requirements and specific applications. For example, in PV connected application, I_p^* is determined by the maximum power point tracking (MPPT) [2], [10]. The phase θ_g is obtained via a phase-locked loop (PLL). To exclusively feed the grid with pure active power, ϕ is set to zero; however, it is adjusted as needed to meet the grid's reactive power demands. Hence,the second control objective can be achieved by minimizing e_i.

It is worth noting that the two control objectives are interrelated; they are dependent on each other, and any control action will affect both objectives. The relationship between V_2 and i_g can be understood from the mathematical model of the system, as explained in [35].

III. REINFORCEMENT LEARNING-BASED CONTROL FOR PUC-MLI

This section provides a detailed explanation of the proposed RL-C methodology. First, the overall framework is presented, followed by a formal definition of the Markov Decision Process (MDP). The chosen RL algorithm, PPO, is then described, and finally, the specific learning setup for the PUC-MLI system is outlined.

A. REINFORCEMENT LEARNING FRAMEWORK

The proposed RL-C, visually represented in Fig. 2, comprises two key phases: an offline training phase and an online implementation phase. The training phase is implemented in simulation using MATLAB/Simulink where the RL agent interacts with the PUC5 environment. The agent observes the system's state, and based on this state observation, the agent

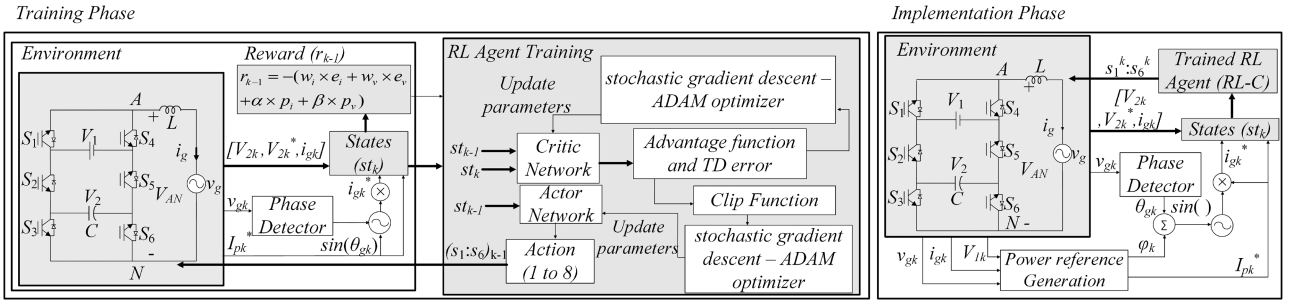

FIGURE 2. The proposed reinforcement learning based controller for PUC5 system in the training phase and the implementation phase.

TABLE II RL-C Training Parameters

Parameters	Value	Parameters	Value
Sampling time (T_s)	$20\mu s$	I_l	0.05
Fundamental frequency	$60Hz$	V_l	0.1
Grid Voltage v_g (peak)	170 V	α and β	0.5
Filtering inductor L	$5mH$	w_v	0.3
PPO clipping factor	0.2	w_i	1
RL discount factor (γ)	0.95	Capacitor C	$0.5mF$
PPO mini-batch size	200	critic learning rate	0.0008
Number of iterations/episode	1660	actor learning rate	0.0008

selects an action, corresponding to a specific switching configuration of PUC5, without the need for a modulation step as in other designs. The environment then transitions to a new state, and the agent receives a reward signal that quantifies the desirability of the chosen action. This iterative process allows the agent to learn an optimal policy, which maps states to actions, by maximizing the cumulative reward over time. PPO algorithm is employed to facilitate this learning process.

The implementation phase, carried out in simulation or experimentally, involves deploying the trained RL agent to control the physical PUC5 system. In this phase, the agent observes the real-time system states and, using its learned policy, determines the appropriate switching actions which are then applied to the inverter, effectively regulating its output voltage and current. This closed-loop control system allows the trained agent to respond dynamically to changing operating conditions and reference signals. The reference signals are generated based on the application and grid requirements.

B. RL-C PROBLEM FORMULATION

The RL-C MDP-based formulation consists of designing the environment, the observed states, the control action, and the reward function.

- *Environment:* The environment in this study is defined by the grid-connected PUC5 system, which captures the system dynamics governed by the inverter's electrical characteristics and variations in the operating states, as shown in Fig. 2. The PUC5 system parameters used during the training phase are summarized in Table 2.

The training environment is designed to expose the proposed RL-C to diverse operating conditions, specifically varying current and voltage references that correspond to changes in the grid-fed current and the DC voltage source. This setup enables the agent to learn a generalized control policy for the grid-connected PUC5 system. To achieve this, reference values for V_2^* and i_g^* are generated by varying the input parameters V_1 and I_p^* within the ranges of 195V to 210V and 3A to 15A, respectively. In each training epoch, two random combinations of these reference values are selected using a uniform distribution. Each combination is applied for one cycle, resulting in two cycles per epoch. During each time step (k), the environment interacts with the agent by providing state observations and receiving corresponding control actions. The agent is trained under fixed grid voltage and frequency conditions, and a unity power factor is assumed. These grid parameters are held constant during training and are only varied during testing to evaluate the generalization performance of the trained controller.

- *State space:* The state space, denoted by $\mathcal{S}$, encapsulates the observed states from the environment that required to characterize the system's operating condition. The state $st_k \in \mathcal{S}$ is defined as a vector comprising the following measurements:

$$st_k = [V_{2k}, V_{2k}^*, i_{gk}, i_{gk}^*, I_{pk}^*] \tag{4}$$

Consequently, the state space is a continuous 5-dimensional space. Compared to the model proposed in [10], this design reduce the state space by eliminating v_g state from the RL-C design, which is a scaled signal of i_g^* in this training design.

To ensure numerical stability and efficiency during training, all states are normalized. Specifically, all voltage states are normalized by 100, while all current states are normalized by 5. This normalization ensures that the input space is appropriately scaled for the learning algorithm, preventing issues associated with large or inconsistent input magnitudes. Moreover, it is worth noting that the observed states are identical to those required by other control algorithms applied to the PUC5 system, indicating that the RL-C does not require any additional measurements from the system.

- *Action space:* The action space, denoted by $\mathcal{A}$, comprises a finite set of discrete control actions corresponding to the permissible switching configurations of PUC5. As enumerated in Table 1, eight distinct switching configurations are feasible. Formally, the action space can be written as $\mathcal{A} = a_1, a_2, \ldots, a_8$, where a_i represents the i-th switching configuration. At each time step k, the RL agent selects an action $a_k \in \mathcal{A}$ based on its policy, thereby influencing the system's dynamics.
- *Reward function:* The reward function, denoted by r, serves as a performance metric for the RL agent, quantifying the desirability of its actions in a given state. The objective is to track the reference signals, while adhering to operational constraints. Hence, the reward function is formulated as a combination of tracking errors and penalty terms:

$$r_k = -w_i \times e_{ik} - w_v \times e_{vk} - p_{ik} - p_{vk}, \tag{5}$$

where w_i and w_v are weighting factors for the tracking errors, and p_{ik} and p_{vk} are penalty terms for violating current and voltage constraints, respectively, defined as:

$$p_{ik} = \begin{cases} \alpha & \text{if } i_{gk} > (1 + I_l) I^*_{pk}, \\ \alpha & \text{if } i_{gk} < -(1 + I_l) I^*_{pk}, \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

$$p_{vk} = \begin{cases} \beta & \text{if } V_{2k} > (1 + V_l) V^*_2 \\ \beta & \text{if } V_{2k} < (1 - V_l) V^*_2 \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

where α and β are the penalties magnitudes, and I_l and V_l are the current and voltage bounding limits. The aforementioned formulation of the reward function incentivizes the RL agent to minimize tracking errors while penalizing constraint violations.

C. PROXIMAL POLICY OPTIMIZATION (PPO) ALGORITHM

PPO is an RL algorithm that belongs to the family of actor-critic methods, as it simultaneously optimizes both a policy (actor) and a value function (critic) to ensure stable and efficient learning. Accordingly, the RL architecture in the proposed RL-C adopts an actor-critic design, with two separate NNs developed for the actor and critic, respectively.

Unlike other policy gradient methods, PPO controls the magnitude of policy updates through a clipped objective function, ensuring stability during training [37]. In each iteration, the agent selects actions based on its current policy, interacts with the environment, and receives feedback in the form of rewards, which are used to improve future decisions. The PPO method naturally maintains a balance between exploration (trying new actions) and exploitation (refining actions that yield higher rewards) by limiting drastic changes to the policy. The clipping mechanism prevents excessively large policy updates, which could prematurely favor exploitation of suboptimal actions, while still allowing the agent to explore new actions and gradually improve its policy [38].

The block diagram of the PPO architecture and training algorithm is illustrated in the training phase in Fig. 2, and Algorithm 1 presents a detailed step-by-step procedure for training the PPO. The agent observes the state st_{k-1} and feeds it to the actor network, which provides the action a_{k-1} based on its current policy, and to the critic network that predicts the expected return (value val_k). This action is applied to the PUC5 system, and the resulting state st_k and the immediate reward r_k are sent back to the agent as feedback. The difference between the actual accumulated discounted future rewards (R_k) (the return) and the predicted value from the critic network defines the advantage function A_k as follwos:

$$A_k = R_k - val(st_{k-1}) \tag{8}$$

$$R_k = \sum_{n=0}^{N} \gamma^n r_{k+n}, \tag{9}$$

where γ is a discount factor and N is the prediction horizon. A positive A_k indicates that the actual return is higher than the predicted value of the state, meaning the action performed better than expected and the probability of selecting this action is increased. Conversely, a negative A_k indicates that the action performed worse than expected, and the probability of selecting this action is reduced. Accordingly, the probability ratio (pr_k) is defined to compare the likelihood of selecting action a_k for an observed state st_k under the updated policy relative to the previous policy:

$$pr_k = \frac{\pi_{\theta_a}(a_k \mid st_k)}{\pi_{\theta_a \text{old}}(a_k \mid st_k)}, \tag{10}$$

where π_{θ_a} and $\pi_{\theta_a \text{old}}$ are the current policy and the previous policy, respectively.

The computed A_k and pr_k define the actor objective function, which is expressed as:

$$L_k^{\text{actor}} = \min\left(pr_k \cdot A_k,\ \text{clip}\left(pr_k,\ 1 - \epsilon,\ 1 + \epsilon\right) \cdot A_k\right), \tag{11}$$

where ϵ is the clip function hyperparameter that defines the allowable range for policy updates, ensuring that changes between successive policies remain moderate to maintain stable learning. The actor loss function, used to update the actor NN weights, is the negative of L_k^{actor}. On the other hand, the critic NN loss function is defined as the difference between the critic's predicted value and the actual observed return:

$$L_k^{\text{critic}} = \left(R_k - val(s_k)\right)^2. \tag{12}$$

D. PPO LEARNING SETUP

In the proposed RL-C, the critic network consists of three fully connected layers with ReLU activation functions. The network has 50 neurons in the first hidden layer and 25 neurons in the second hidden layer. The actor network, also consists of three fully connected layers with ReLU activations and a softmax layer at the output to represent the action probabilities. The actor network has 80 neurons in the first hidden layer

Algorithm 1: PPO Training Algorithm.

Initialize the actor $\pi(a|st;\theta_a)$ and the critic $val(st;\theta_c)$ with random parameter values θ_a and θ_c, respectively.
Initialize the experience buffer
while training episode not terminated **do**
  Generate N experiences by based on current policy: $[st_n, a_n, r_{n+1}, st_{n+1}]_{n=0}^{N}$
  For each experience sequence from step n $= t_s$ **to** $t_s + N - 1$:
  Compute the TD errors δ_k for each time step:

$$\delta_k = r_{k+1} + b \cdot \gamma \cdot val(st_{k+1};\theta_c) - val(st_k;\theta_c)$$

  Compute the advantage A_n (8)
  Compute the return R_n (9)
  For each learning epoch (K epochs):
  for epoch = 1 to K **do**
    Sample a random mini-batch (size M) from the experiences
    Update critic parameters by minimizing the critic loss (12)
    Update actor parameters by minimizing the actor loss (11)
  end for
end while

and 40 neurons in the second hidden layer. Both networks are optimized using the Adam optimizer, with learning parameters shown in Table 2. The number of iterations per episode is set to 1660 to ensure two complete sinusoidal cycles within each episode, while the batch size is set to 200 to represent a quarter cycle of the sinusoidal signal. The discount factor is chosen as 0.95, based on its demonstrated effectiveness in previous related work [10]. Regarding the penalty values, a higher penalty is assigned to the current error to prioritize it over the voltage error, which is more critical in grid-connected applications.

IV. SIMULATION RESULTS

This section presents a comprehensive evaluation of the proposed RL-C approach, followed by a comparative analysis with both traditional methods and RL-based techniques.

A. PERFORMANCE OF THE PROPOSED RL-C

The training curve, which displays the averaged rewards for every 25 episodes and a zoomed-in view on the average reward for every five episodes (using a moving average window), is depicted in Fig. 3. Due to the exploration/exploitation behavior inherent in RL training, the averaged learning curve is typically studied rather than analyzing the reward on an episode-by-episode basis. Fig. 3 demonstrates that the training stabilized towards the end of the episodes. Initially, the minimum obtained reward was -8.14×10^5, which improved significantly, reaching a maximum reward of -283.3. This

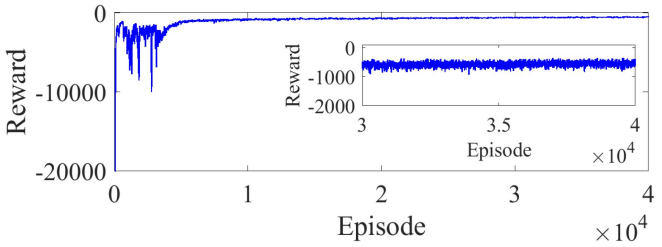

FIGURE 3. The learning curve of the proposed RL-C.

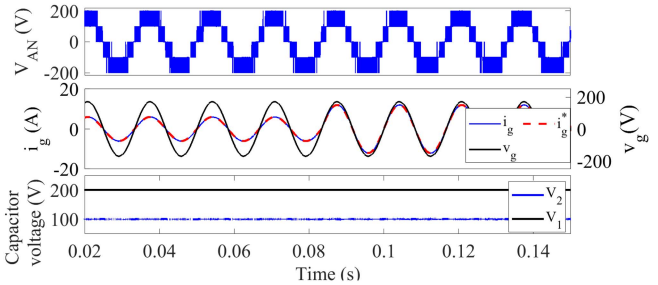

FIGURE 4. The resulted steady-state performance of PUC5 connected to 170 V, 60 Hz grid in the testing phase.

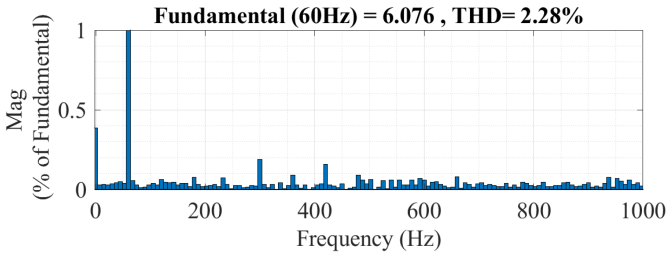

FIGURE 5. The frequency spectrum of the generated current with 6 A current magnitude.

agent, achieving the maximum reward, is considered the best-trained agent and is utilized in the system implementation, as illustrated in Fig. 2, and to test the proposed RL-C design both in simulation and experimentally.

The trained agent was tested with smaller L and C sizes, differing from the training parameters listed in Table 2; specifically, $L = 4$ mH and $C = 100$ μF, where the PUC5 is connected to a grid with 170 V peak voltage and a frequency of 60 Hz. The performance of the PUC5, controlled by the RL-C, is shown in Fig. 4, where i_g accurately tracked i_g^* with I_p^* varying from 6 A to 12A. Furthermore, V_2 was regulated at its reference value, resulting in the generation of five DC levels at V_{AN}. The THD of the generated current at 6 A was found to be 2.28%, with the spectrum depicted in Fig. 5. Fig. 6 shows the current THD at 10 A, which was found to be 1.4%.

B. COMPARISON WITH TRADITIONAL APPROACHES

The performance of the proposed RL-C is compared with four control algorithms applied to the PUC-MLI [3], [5], [6], [35].

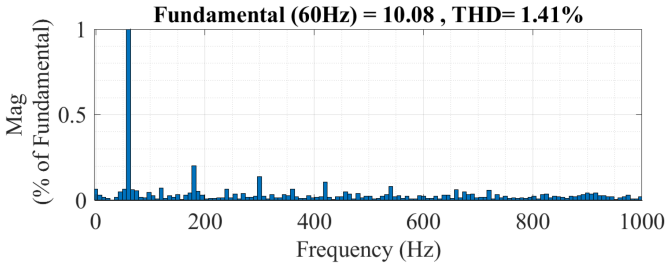

FIGURE 6. The frequency spectrum of the generated current with 10 A current magnitude.

The controller proposed in [3] represents the simplest control design, consisting of two proportional-integral (PI) control loops: the outer loop regulates the capacitor voltage, and the inner loop controls the current. The controller in [5] is an FCS-SMC, which includes two sliding functions, one for each control objective. At each time step, this controller evaluates the control law for all switching patterns and selects the state that satisfies the sliding surface conditions, hence referred to as SMC-online. This model requires a predefined hysteresis bandwidth for the capacitor voltage, set at 1V [5]. Conversely, the SMC proposed in [6] relies on a predetermined lookup table calculated offline by considering all possible operating conditions alongside the sliding surface conditions to accelerate control action selection through direct operational state references based on voltage and current errors. This controller is referred to as SMC-offline. Another effective controller is the FCS-MPC, applied to the PUC5 for its high performance in reducing current THD and minimizing capacitor voltage error simultaneously, using a weighted single cost function that combines the two control objectives as proposed in [35]. Optimal performance is achieved when the current objective weighting factor is set to 1 and the capacitor voltage objective weight is set to 0.2. The performance of these four controllers, along with the proposed RL-C, is depicted in Fig. 7 across five different L and C sizes in PUC design. The performance metrics considered include current THD, mean absolute error (MAE) of V_2 (MAE(e_v)), and mean switching frequency (f_{sw}), measured for a range of $|i_g^*|$ from 4 A to 16 A in 1 A steps, with average metrics calculated over ten cycles for each case.

The results in Fig. 7 demonstrate that the proposed RL-C generated i_g with a THD comparable to that of the FCS-MPC, but with a slightly higher MAE(e_v) and a lower switching frequency (f_{sw}) across all tested scenarios. Specifically, the MAE(e_v) was maintained below 2 V with 0.5 V mean, and f_{sw} averaged around 15 kHz. Compared to the PI, SMC-online, and SMC-offline controllers, the RL-C achieved notably lower THD across the tested range of $|i_g^*|$, particularly excelling at lower current magnitudes where the other controllers tended to perform better at higher magnitudes. The 5 mH, 500 μF case, used for the training of the RL-C, alongside other cases not included in the training, illustrates the generalization capability of the proposed AI-based controller. This adaptability demonstrates its potential applicability to other PUC systems without the need for retraining. Furthermore, the exceptional performance of the RL-C under variable current references highlights its suitability for systems with dynamic operating conditions, such as those encountered in renewable energy systems.

Since the performance of the proposed RL-C closely approaches that of the FCS-MPC, their respective responses to changes in V_2^* were analyzed for a configuration with $L = 5$ mH and $C = 500$ μF and $|i_g^*| = 10$ A case. As depicted in Fig. 8, V_1 underwent a step-up from 200 V to 260 V over five cycles, inducing a V_2^* variation from 100 V to 130 V. The results presented in Fig. 8 demonstrate that the model-free RL-C responds as rapidly as the FCS-MPC to variations in voltage references. Specifically, the RL-C recorded steady-state voltage MAEs of 0.2 V and 0.9 V at 100 V and 130 V, respectively, while the FCS-MPC showed MAEs of 0.1 V and 0.7 V at the same reference voltages. On the other hand, unlike the model-free PI controller, the RL-C maintained a smooth generated current and accurately tracked the reference signal without distortion during the voltage step change. For the PUC5 system, the FCS-MPC determines its control action within 0.5 μs, while the proposed model-free RL-C requires 1.7 μs for action inference. Given that the controller's sampling period is 20 μs, both methods operate well within the real-time constraints and do not suffer from latency issues.

C. THE MODEL FREE RL-C VS FCS-MPC UNDER MODEL MISMATCH

The proposed model-free RL-C controller does not rely on the values of passive components to determine control actions, as detailed in Section III. The agent is trained using a specific configuration and evaluated under varying system parameters, as illustrated in Fig. 7, demonstrating its generalization capability. In contrast, the model-based FCS-MPC controller depends on an accurate mathematical model and precise knowledge of component values.

This section compares the performance of both controllers under component mismatch conditions. While the FCS-MPC algorithm assumes ideal component sizes, deviations are introduced into the PUC5 hardware configuration to simulate modeling inaccuracies or aging effects. To isolate the impact of each component, the filtering inductor and auxiliary capacitor values are varied separately in the analysis. That is, the inductor value is held constant while varying the capacitor, and vice versa. Two test scenarios are considered, as shown in Figs. 9 and 10. Both controllers maintain a the MAE(e_v) below 1 V across the tested mismatch ranges. However, RL-C consistently achieves lower THD, particularly under reduced capacitor values. Additionally, both controllers exhibit greater sensitivity to inductor variation than to capacitor mismatch.

Furthermore, to highlight the difference between FCS-MPC and the proposed RL-C in handling component mismatch, the performance of both controllers under an extreme case in which the inductance is physically reduced by 50% (from 8 mH to 4 mH) while the system is operating, are shown in

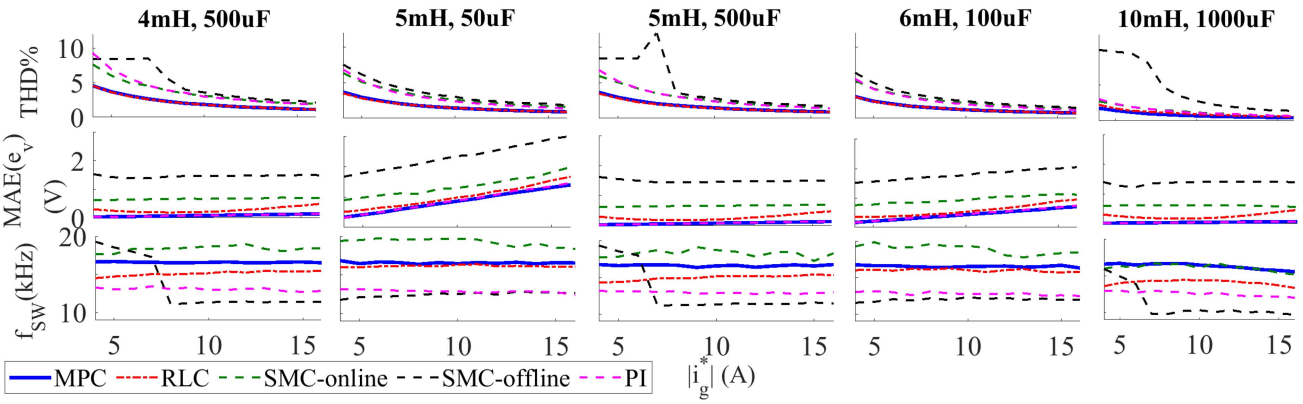

FIGURE 7. RL-C performance comparison with traditional approaches: MPC, SMC online, SMC offline, and PI. The resulted current THD, capacitor voltage MAE and switching frequency of different control algorithms for the different PUC5 designs under a range of operating current magnitude.

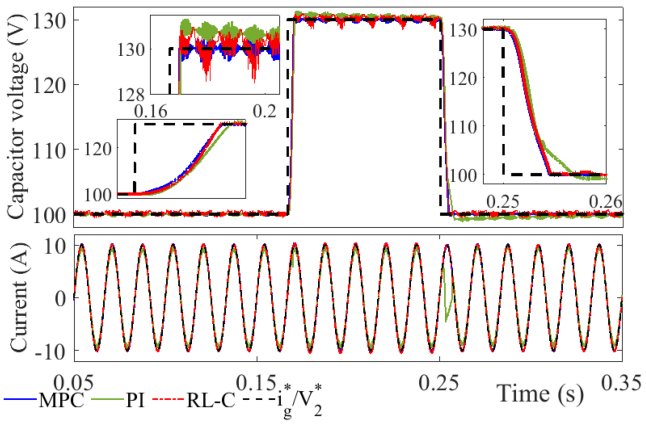

FIGURE 8. The dynamic performance of RL-C, MPC and PI under 30 V step change in V_2.

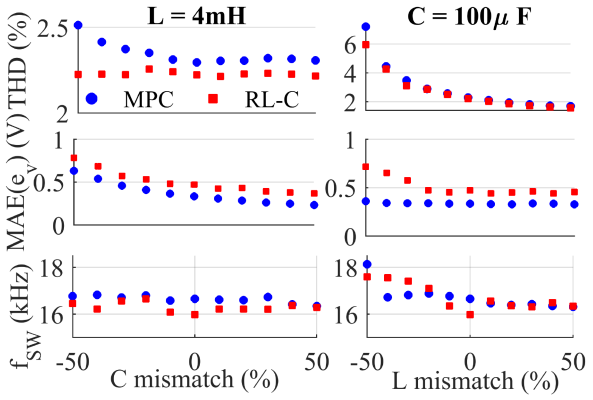

FIGURE 10. The RL-C and FCS-MPC under component ±50% mismatch for 4 mH inductor and 100μF capacitor and 8 A grid current.

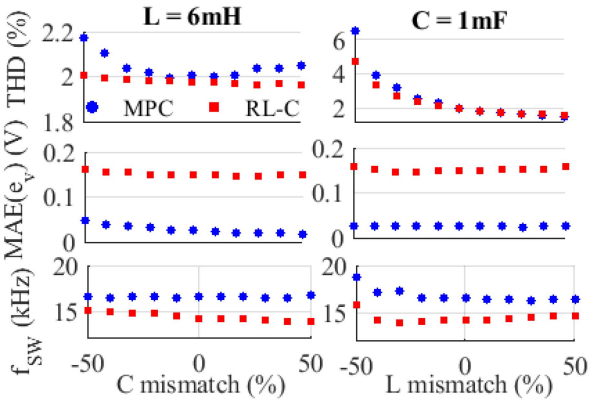

FIGURE 9. The RL-C and FCS-MPC under component ±50% mismatch for 6mH inductor and 1mF capacitor and 6 A grid current.

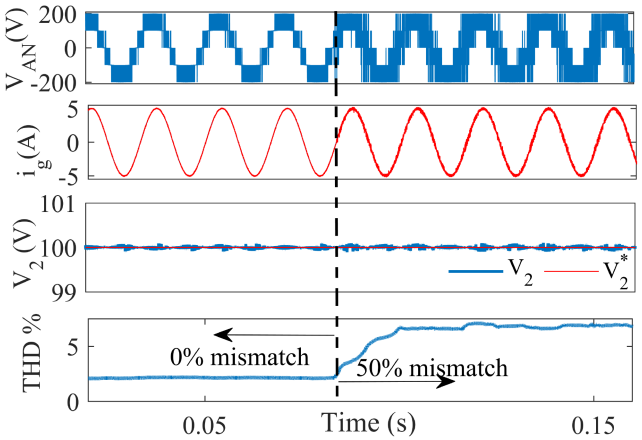

FIGURE 11. FCS-MPC performance under −50% inductance mismatch.

Figs. 11 and 12, respectively. The model-free RL-C maintained the current THD below 5% and regulated the capacitor voltage to preserve the five voltage levels at the inverter output, whereas, the FCS-MPC produced a high current THD under the component mismatch and generated a distorted five-level output voltage waveform.

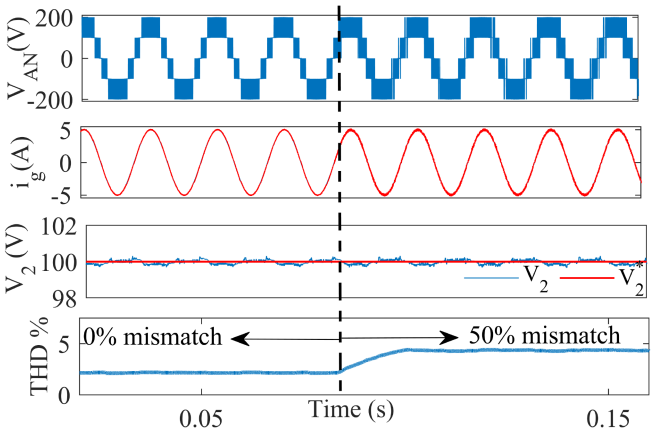

FIGURE 12. RL-C performance under −50% inductance mismatch.

TABLE III The Average of One Episode Simulation Time and Saved Agent File Size

	PPO	DQN	DDQN	TRPO	PPOs
Episode time (s)	3.74	19.13	19.27	18.83	3.69
File size (kB)	44	26	26	46	35
Online computation time (μs)	1.7 ±0.01	1.2 ±0.02	1.2 ±0.02	2.1 ±0.02	1.4 ±0.01

D. COMPARISON WITH RL-BASED APPROACHES

The performance of the proposed RL-C using the PPO algorithm is compared with three other RL algorithms: DQN, DDQN, and Trust Region Policy Optimization (TRPO). The results obtained from the four models are presented in Fig. 13, while the average time required to complete one episode on the same computer (an Intel Xeon CPU with 128 GB RAM and no dedicated GPU) and the memory size required for a single agent, are summarized in Table 3. The averages are calculated over 10 episodes, with each episode consisting of 833 × 2 iterations.

All three RL designs are applied to the formulated optimization problem, where the observed states are continuous and the action space is discrete, represented by the switching states. DQN is the simplest deep RL algorithm, utilizing a single NN, followed by DDQN, which employs two NN. Both methods have low computational complexity and minimal memory requirements during training. However, these two algorithms are value-based methods with implicit policy optimization, where the Q-value corresponding to each action given the observed state is estimated, and the action with the highest Q-value is selected. As shown in Fig. 13, although DQN achieved low mean voltage error (e_v), its resulting current THD and average switching frequency are higher than those of the proposed PPO model. On the other hand, TRPO, a stochastic policy-based algorithm with explicit optimization and an actor-critic architecture comprising separate value and policy networks, achieved performance comparable to PPO. However, it requires more memory during training due to storing full trajectories and the computational overhead of second-order optimization, as shown in Table 3. This comparison highlights the effectiveness of the PPO algorithm for the system under study, striking a balance between training efficiency, control performance, and memory requirements. The performance of the proposed RL-C using the PPO algorithm comes with a slight increase in online computational time—measured at 1.7 μs as shown in Table 3, compared to 1.2 μs for both DQN and DDQN.

E. RL-C DESIGN SENSITIVITY ANALYSIS

This section presents an analysis of how changes in the reward function, observed states, and neural network size affect the performance and the training phase of the RL-C, aiming to provide optimal guidance for control designers in MLI systems. Table 4 summarizes the sensitivity analysis cases considered in this section.

1) COMPACT NEURAL NETWORK

To investigate the impact of NN size, the proposed compact and shallow architecture, originally designed with two hidden layers, was reduced by 25% in terms of the number of neurons per layer. The original RL-C model features two hidden layers for both the actor and the critic, with 80 and 40 neurons for the actor, and 50 and 25 neurons for the critic. With the 25% reduction, the actor NN consists of 60 and 30 neurons, while the critic is configured with 38 and 19 neurons. This reduced model is referred to as PPOs.

Despite the downsizing, the PPOs model achieved nearly the same current THD and f_{SW} as the original PPO design. However, the mean voltage error (e_v) slightly increased, indicating a modest reduction in voltage regulation performance, as illustrated in Fig. 13. Training time remained largely unaffected, with only a minor improvement in online computational time which is reduced by 0.3 μs, as shown in Table 3.

2) LEARNING RATE

To evaluate the effect of a faster learning rate, the learning rate of the original PPO model was increased by 25% to 0.001, which is commonly used in AI applications. The performance of this modified model, referred to as PPOf, is also presented in Fig. 13. Similar to PPOs, the PPOf model maintained comparable current THD and f_{SW} values to the original PPO design, but with a higher mean e_v.

3) STATE SPACE DOMAIN

The proposed RL-C in this paper is designed with five observed states in the state space, as described in Section III-B. In this section, the impact of reducing the state space to four states, $[V_2, V_2^*, i_g, i_g^*]$, by removing the reference current peak I_p^*, is investigated. The RL-C was retrained using the same environment and model parameters, and this configuration is referred to as PPO4 s.

The performance of PPO4 s is shown in Fig. 14, which demonstrates a slightly higher current THD and a slightly MAE(e_v) compared to the original RL-C (PPO model in

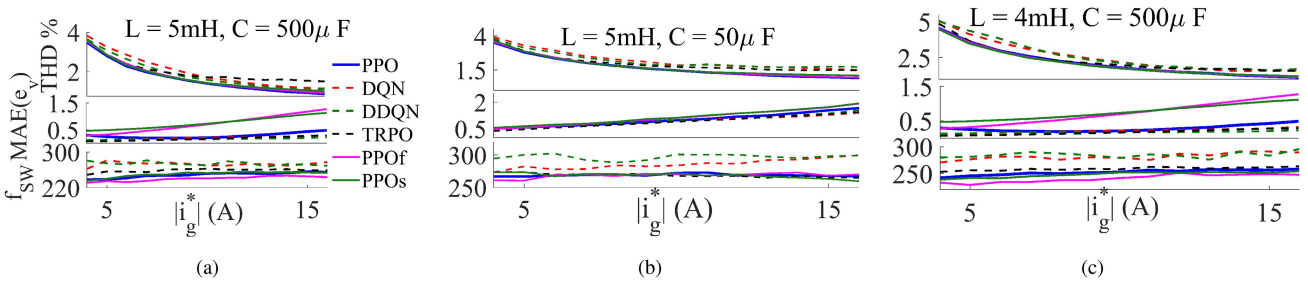

FIGURE 13. Performance comparison of RL-C designs using four RL algorithms (PPO, DQN, DDQN, and TRPO), along with a PPO variant using a smaller neural network and faster learning rate. Results are shown for: (a) the training PUC5 design, (b) a design with reduced *C*, and (c) a design with reduced *L*.

TABLE IV Baseline PPO Configuration and Sensitivity-Analysis Variations

Category	Baseline (PPO)	Sensitivity variation
State space	$[V_2, V_2^*, i_g, i_g^*, I_p^*]$	Reduced (PPO_{4s}): $[V_2, V_2^*, i_g, i_g^*]$; Augmented (PPO_{act}): $[V_2, V_2^*, i_g, i_g^*, I_p^*, a_{k-1}]$
Actor/Critic NN	Actor (80,40), Critic (50,25)	25% smaller (PPO_s): Actor (60,30), Critic (38,19); Deeper critic (PPO_c): Critic (100,50,25)
Learning rate	8×10^{-4}	Increased learning rate (PPOf): 10^{-3}
Discount factor	$\gamma = 0.95$	$\gamma \in \{0.99, 0.95, 0.90, 0.85, 0.75\}$
Reward weights	$w_i = 1,\ w_v = 0.3$	$w_v \in \{0.1, 0.3, 0.5, 0.7, 1\}$, with $w_i = 1$
Penalty terms	Two penalties, $\alpha = \beta = 0.5$	No penalties: $p_i = p_v = 0$; Single penalty: $\alpha = 0$ (only $p_v = 0.5$)

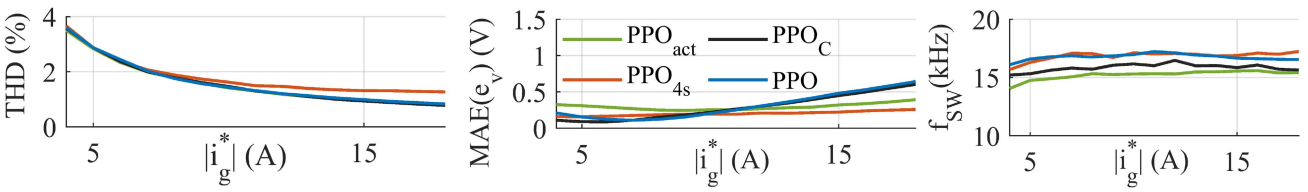

FIGURE 14. Performance comparison of the proposed RL-C design with a reduced state space configuration, state-space with action feedback configuration, and a deeper critic network.

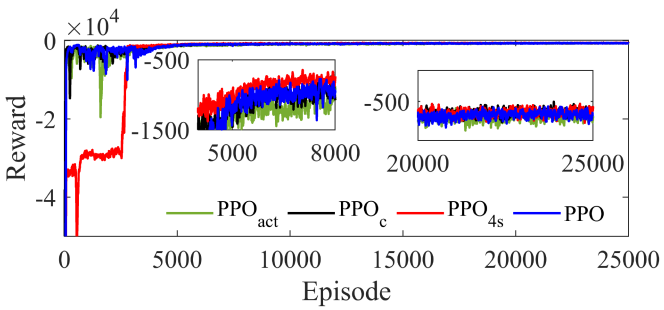

FIGURE 15. Learning curves of the proposed RL-C design compared with the reduced state space configuration, state-space with action feedback configuration, and the deeper critic network variant.

Fig. 14). The reduced state space led to a marginal decrease in average training time, 3.53 seconds per episode compared to 3.74 seconds for the proposed PPO model. However, this small gain in training time came at the cost of slower learning performance. As illustrated in Fig. 15, removing I_p^* from the observed state increased the exploration phase and delayed training convergence.

Furthermore, the impact of including the previous switching state (action at time step $k-1$) as part of the state-space at time step k is investigated. Hence, the state-space in this case consists of six observations. This configuration is denoted as PPO_{act}. As shown in Fig. 14, PPO_{act} did not lead to improvements in current THD but resulted in a slight reduction in switching frequency. The average training time for this case was 4.65 seconds, with a minor delay in convergence, as illustrated in Fig. 15.

4) CRITIC TO ACTOR SIZE

The results presented in this paper are based on an RL-C design using an Actor–Critic architecture, where both networks consist of two hidden layers with (80, 40) neurons for the actor and (50, 25) neurons for the critic. The performance of a smaller overall network, denoted as PPO_s, is shown in Fig. 13, where the entire architecture was reduced by 25%, while maintaining a larger actor network than the critic. Also, the impact of increasing the critic network size on the RL-C performance is investigated, and this configuration is referred to as PPO_c. Specifically, the critic is expanded to three hidden layers with 100, 50, and 25 neurons, respectively, while the

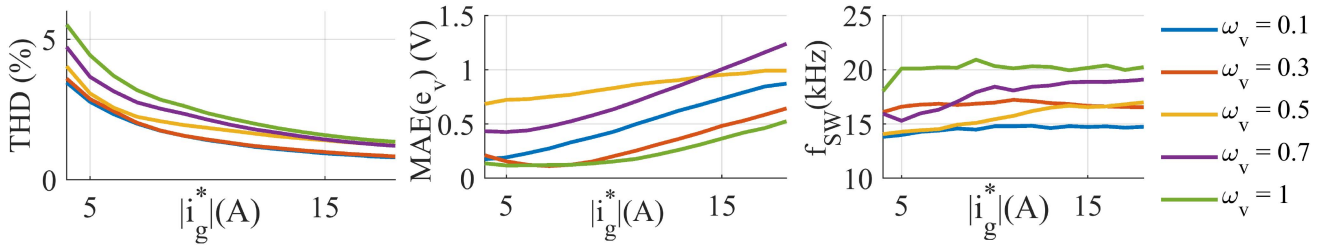

FIGURE 16. The resulted performance of different weighting factor value in the reward function where w_v takes vales of 0.1, 0.3, 0.5, 0.7 and 1 and w_i is fixed to 1.

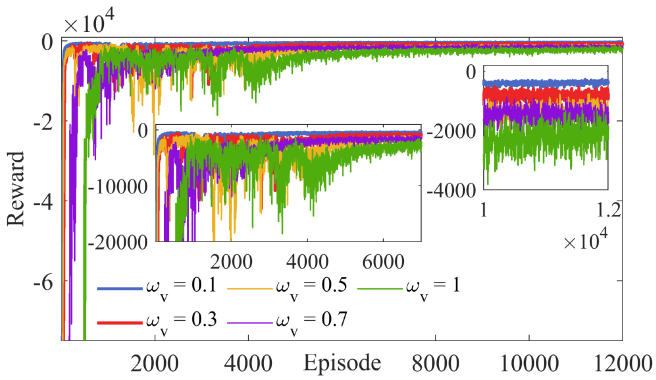

FIGURE 17. The learning curve of different weighting factor value in the reward function where w_v takes vales of 0.1, 0.3, 0.5, 0.7 and 1 and w_i is fixed to 1.

actor network remains unchanged from the original RL-C design.

Increasing the critic network size did not lead to improvements in current THD or the MAE of e_v, as shown in Fig. 14. However, it resulted in a slight enhancement in switching frequency. In terms of learning performance, the deeper critic network introduced greater variability during the exploration phase compared to the original PPO model, as illustrated in Fig. 15, and led to a slower training process, with a mean episode training time of 4.1 seconds, compared to 3.74 seconds in the original configuration. These findings suggest that the added complexity in the critic network does not offer significant performance benefits, and the original compact design remains more efficient.

5) REWARD FUNCTION WEIGHTING FACTORS

The impact of assigning different importance levels to the current and voltage errors in the reward function (5) is investigated in this section. The weighting factor for the current error, w_i, is fixed at 1, while the voltage error weighting factor, w_v, is varied across the values 0.1, 0.3, 0.5, 0.7, and 1. All RL-C models are trained under the same environment and system parameters.

The resulting performance of each configuration is shown in Fig. 16, and the corresponding learning curves are presented in Fig. 17. The results in Fig. 16 indicate that setting w_v to higher values—thus giving equal importance to current and voltage objectives—leads to a higher THD across the studied operating range. When $w_v = 1$, the model achieves the lowest MAE(e_v), but this comes at the cost of significantly increased THD and the highest switching frequency.

For a grid-connected application, where prioritizing current control is typically more critical, the best performing configurations are those with $w_v = 0.1$ and $w_v = 0.3$. Both result in MAE(e_v) values below 1 V. However, $w_v = 0.3$ yields a slightly lower MAE(e_v), and is thus selected as the model for generating all results in this paper.

From the learning curve perspective, all models eventually converge, but those with smaller w_v values reach steady learning faster than those with larger w_v, which require longer exploration phases. It is important to note that since the reward function varies between cases due to different weightings, the reward values themselves are not directly comparable; rather, the focus is on convergence speed and exploration behavior.

6) REWARD FUNCTION PENALTY TERMS

In this section, the impact of including the penalty terms p_i and p_v in the reward function (5) is investigated. In the first test case, both penalty terms are set to zero ($p_i = p_v = 0$), such that the reward function depends solely on the error terms, without imposing penalties for values outside the bounded operating range. In the second test case, the current penalty term is set to zero ($\alpha = 0$ in (6)), and a single penalty term corresponding to the voltage error, p_v with $\beta = 0.5$, is applied in the reward function as defined in (5). The third case represents the proposed RL-C training model in this paper, with detailed parameters provided in Table 2, where both penalty terms are included with $\alpha = \beta = 0.5$. For all cases, the weighting factors are set to $w_i = 1$ and $w_v = 0.3$.

The results of the three cases are shown in Fig. 18. Including the penalty terms in the reward function does not significantly affect the resulting current THD, with only a slight difference observed in the MAE of e_v. However, it leads to a slightly higher switching frequency compared to training without penalties or with only a single penalty term. On the other hand, incorporating both penalty terms during training results in an RL-C agent that maintains a narrower capacitor voltage range compared to the other two cases.

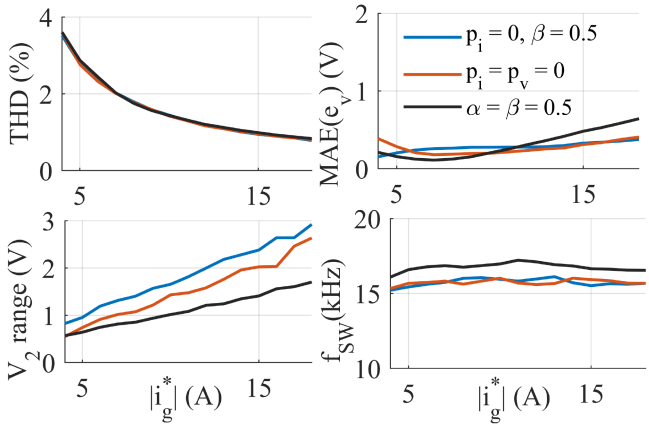

FIGURE 18. The resulting performance of the RL-C models trained with and without incorporating penalty terms in the reward function.

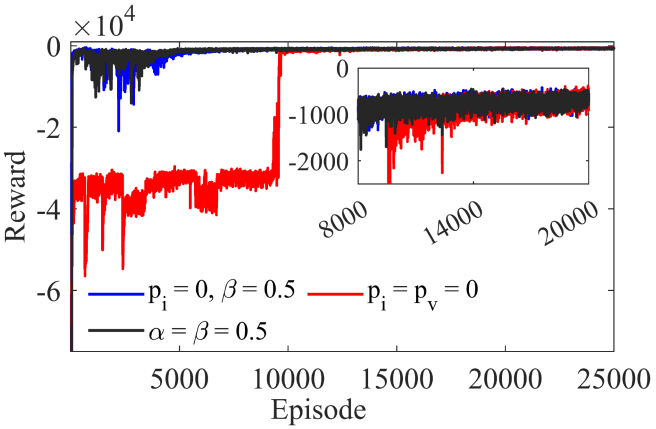

FIGURE 19. The learning curves of the RL-C models trained with and without penalty terms in the reward function.

Overall, the results indicate that incorporating penalty terms in the reward function (5) does not significantly alter the control performance, as all trained agents were able to meet the control objectives. However, the impact of the penalty terms is more evident in the learning curves shown in Fig. 19. Training the RL-C without penalty terms leads to a delayed convergence, with the agent reaching the convergence boundary after approximately 10,000 episodes. In contrast, including penalty terms, which guide the agent to remain within acceptable bounds during training, accelerates the convergence process. Furthermore, using both penalty terms reduces the exploration duration and allows the learning process to stabilize more quickly compared to using only a single penalty term.

Based on the conducted analysis, the results demonstrate that the proposed reward function for the PPO algorithm, designed for the PUC5 system with a higher weighting factor assigned to the current control objective compared to the voltage control objective, and with incorporated penalty terms, proved effective in enhancing both the agent's performance and training efficiency.

7) DISCOUNT FACTOR

The impact of the discount factor γ in the accumulated discounted future rewards R_k (9) on the RL-C performance and training process is investigated. The RL-C was trained using identical environment settings and actor–critic network architecture, with γ values set to 0.99, 0.95, 0.9, 0.85, 0.75.

The results indicate that RL-C training is not highly sensitive to the choice of γ, as the current THD remains similar across all tested values, as shown in Fig. 20. A slight variation was observed in the voltage MAE, where higher values of γ yielded lower voltage error, while lower γ values led to a marginal increase in error. All models converged within 5000 episodes, as illustrated in Fig. 21. However, training with $\gamma = 0.99$ showed delayed convergence, while $\gamma = 0.75$, 0.85 required a longer exploration period. These results suggest that setting the discount factor to 0.9 or 0.95 is appropriate for this class of systems.

V. EXPERIMENTAL RESULTS

The proposed RL-C was tested experimentally in the laboratory using a grid-connected PUC5 prototype (Fig. 22), where a low-cost practical sensors are used for voltage and current measurements. The MATLAB-built RL-C was implemented in OPAL-RT 5700 (for its compatibility with reinforcement learning MATLAB toolbox).

The primary challenge in deploying an offline-trained RL-C agent in a real-time system lies in designing actor–critic neural networks that are sufficiently compact to ensure computation times remain well below the sampling period. This prevents computational delays and avoids overlap between the sampling and action windows, which could otherwise compromise control performance. To address this, the RL-C agent proposed in this work adopts a compact, fully connected architecture with fewer than 5,000 total parameters, making it highly suitable for low-latency inference and efficient FPGA implementation, as demonstrated in recent studies such as [39], [40], [41], [42]. Notably, even recurrent architectures, which are more computationally demanding than the proposed design, have achieved computational times below 2 μs, depending on quantization strategies and resource reuse. The real-time computational burden of the proposed RL-C is measured at 1.7 μs, as reported in Table 3, leaving a safe margin of 18.3 μs within the 20 μs sampling period. This indicates the potential of the RL-C for deployment in systems with even higher sampling frequencies. Reducing the size of the neural networks by 25% yields only a modest improvement in computational time (0.3 μs), but leads to a noticeable increase in voltage error, as discussed in Section IV.

In this experiment, $L = 5$ mH and $C = 100$ μF, which is a different configuration from the one used in the training

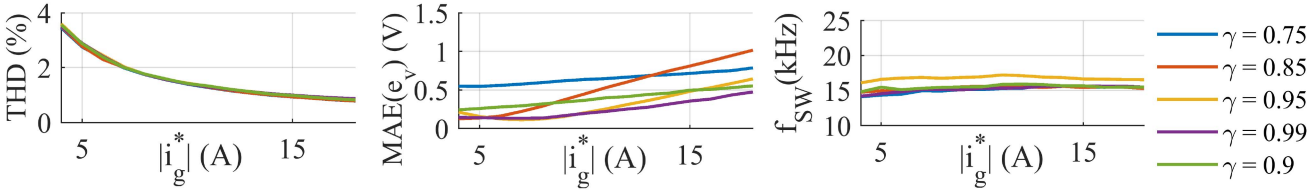

FIGURE 20. The resulted performance of different discount factor value in the return function where γ takes vales of 0.75, 0.85, 0.9, 0.95 and 0.99.

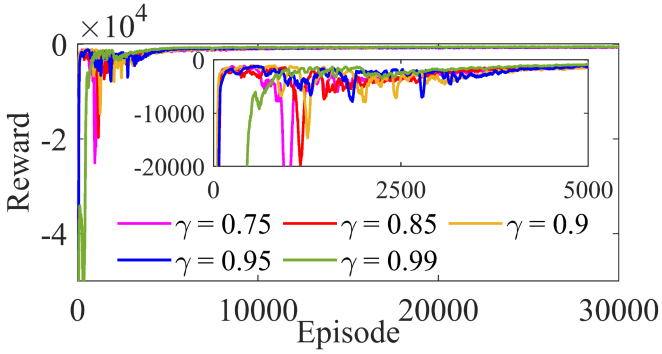

FIGURE 21. The learning curve of different discount factor value in the return function where γ takes vales of 0.75, 0.85, 0.9, 0.95 and 0.99.

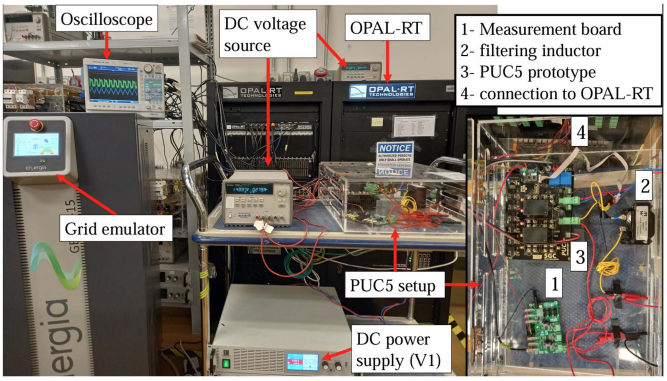

FIGURE 22. The laboratory experimental setup.

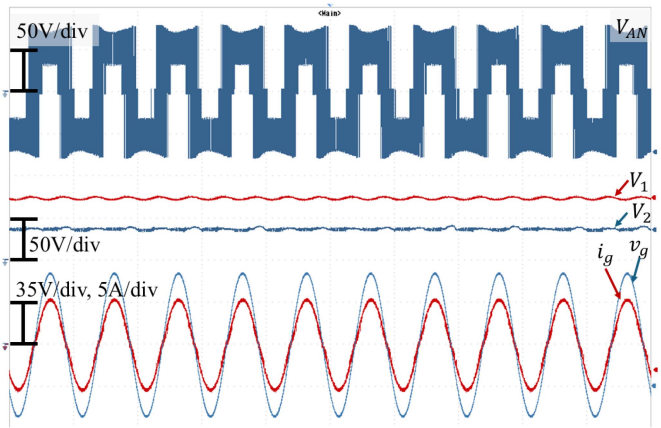

FIGURE 23. The experimental steady state performance of RL-C with $L = 5$ mH, $C = 100$ μF.

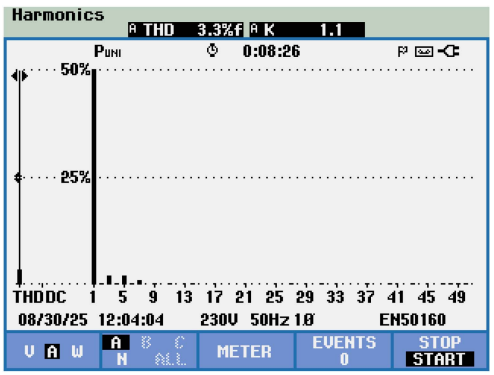

FIGURE 24. The resulted current grid THD at 5 A current reference.

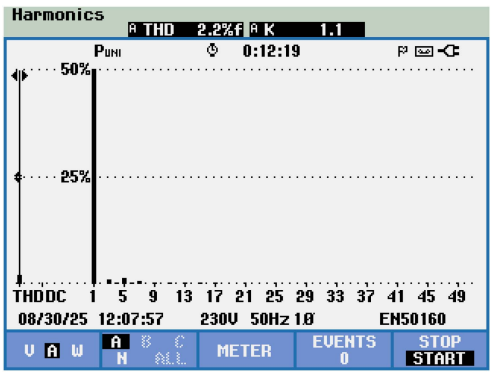

FIGURE 25. The resulted current grid THD at 7 A current reference.

phase, with grid voltage peak set to 60 V with 50 Hz frequency. The DC voltage $V_1 = 75$ V is supplied through a DC power supply to PUC prototype through another capacitor (C_1) with size of 100 μF. The steady-state performance (for $I^*_{gp} = 5$ A) is depicted in Fig. 23. One can notice that the grid current and grid voltage are synchronized, V_2 is regulated around its desired value, and the output voltage V_{AN} shows the 5 expected voltage levels. Figs. 24 and 25 show 3.3% and 2.2% measured current THDs corresponding to the reference currents of $I^*_{gp} = 5$ A and $I^*_{gp} = 7$ A, respectively.

Then, the RL-C performance is investigated under a dynamic current reference signal as shown in Fig. 26, where the current reference, I^*_{gp}, was increased from 4 A to 7 A. The results demonstrate that the proposed RL-C accurately

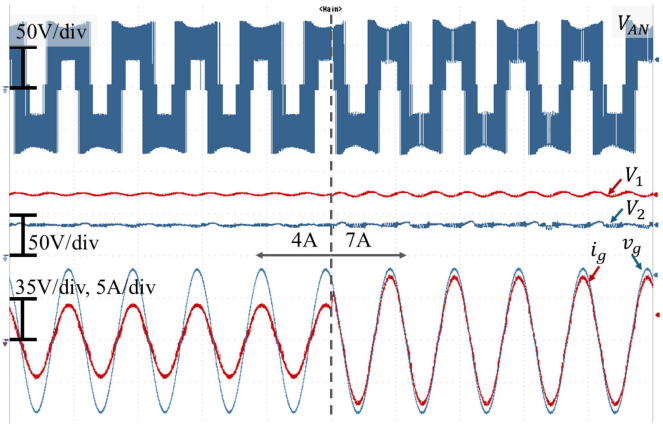

FIGURE 26. Dynamic response of RL-C to current reference change.

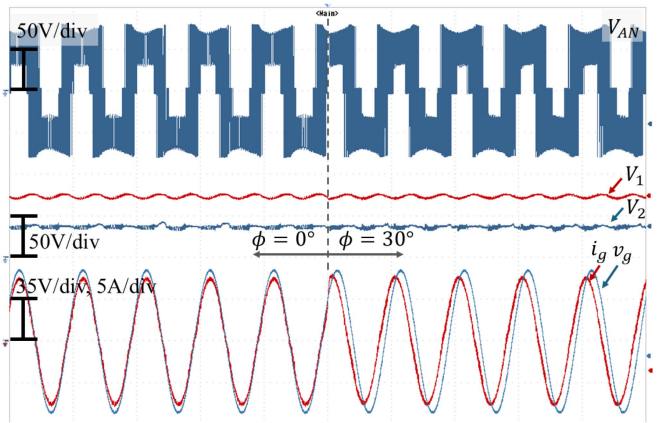

FIGURE 27. RL-C performance under 30^o phase shift between the grid voltage and the grid current.

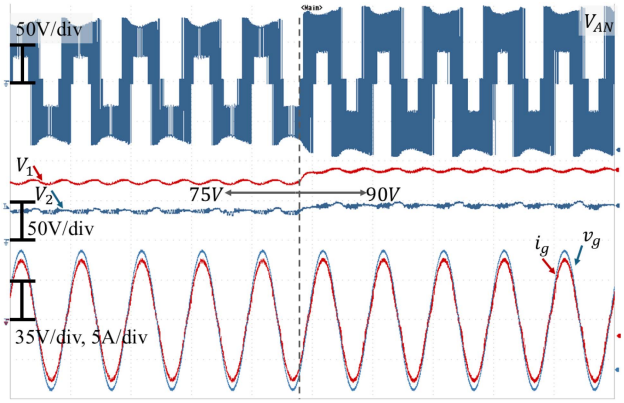

FIGURE 28. Voltage regulation by RL-C during DC source voltage step changes.

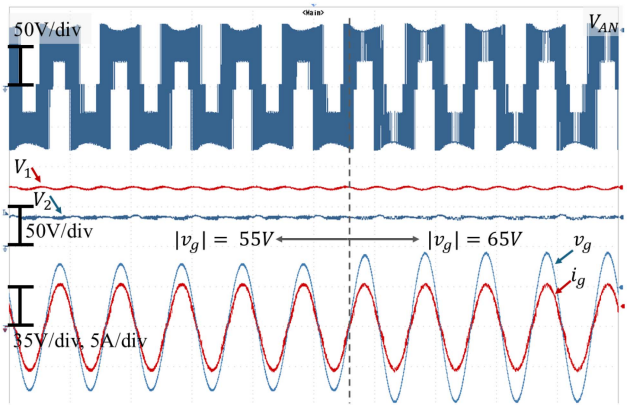

FIGURE 29. RL-C performance under 8% grid voltage change.

tracked the reference change. Although the designed RL-C was initially trained to feed the grid with pure active power (grid synchronization), the capability of the trained agent to also handle reactive power requirements was tested. This was achieved by introducing a phase shift between v_g and i_g^*. As shown in Fig. 27, a $\phi = 30°$ phase shift was implemented, and the RL-C accurately tracked the change in the reference current signal without impacting its performance.

In Fig. 28, the DC voltage source V_1 was stepped up from 75 V to 90 V, illustrating the RL-C effectiveness in regulating V_2 to its new reference value. In addition, the proposed RL-C was subjected to a voltage sag and voltage swell, defined by the IEEE 1159-2019 standard as a variation in voltage magnitude from 0.1 to 0.9 per unit (pu) and 1.1 to 1.8, respectively, for a duration of 0.5 to 30 cycles. This scenario is depicted in Fig. 29 showing the RL-C's performance under a 8.3% grid voltage difference from 60 V. The results demonstrate that the designed controller effectively mitigated this change, maintaining both the regulation of the capacitor voltage and the current reference tracking despite the variation in $|v_g|$. Furthermore, the robustness of the designed RL-C in delivering low THD current to the grid, despite the presence of grid

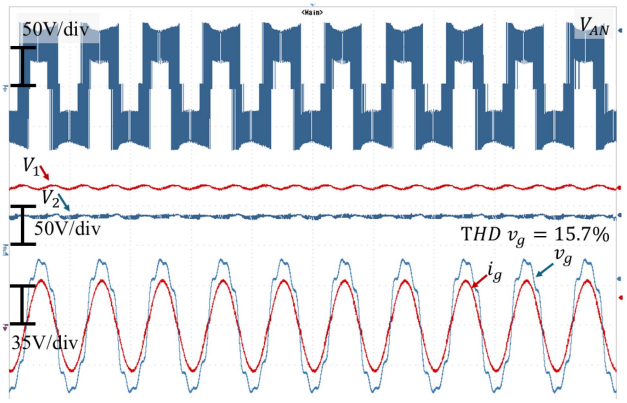

FIGURE 30. RL-C performance under 15.7% grid voltage THD.

voltage harmonics, is demonstrated in Fig. 30, where the grid voltage THD was set to be 15.7% by introducing the 3rd, 5th, and 7th harmonic components to be 10%, 9%, and 8% of the fundamental, respectively.

VI. CONCLUSION

This work proposed a model-free reinforcement learning controller (RL-C) for a grid-connected PUC5 inverter using an actor-critic architecture and PPO algorithm. The controller was trained to operate under various dynamic conditions without relying on a system model or reference controller. Extensive simulations and experimental tests confirmed the RL-C's ability to regulate the PUC capacitor voltage, minimize grid current THD, and respond effectively to dynamic current and voltage references, including grid disturbances and parameter mismatches. Compared to finite control set model predictive control (FCS-MPC), the RL-C achieved similar THD and voltage tracking with lower switching frequency, and outperformed conventional PI and sliding mode control strategies. Moreover, the adaptability of the proposed controller across different operating conditions, without retraining, highlights its strong generalization capabilities. In addition to outperforming conventional and previously proposed controllers, the proposed RL-C demonstrates competitive performance against other RL algorithms. These findings position deep RL as a viable candidate for smart control of next-generation MLIs and power conversion systems. Furthermore, the presented sensitivity analysis provides quantitative insights into how reward formulation, state representation, network size, and hyperparameter selection influence convergence behavior and overall closed-loop performance. This offers a practical guidance for robust controller design.

ACKNOWLEDGMENT

The statements made herein are solely the responsibility of the authors.

REFERENCES

[1] T. Dragičević, S. Vazquez, and P. Wheeler, "Advanced control methods for power converters in DG systems and microgrids," *IEEE Trans. Ind. Electron.*, vol. 68, no. 7, pp. 5847–5862, Jul. 2021.

[2] M. Trabelsi, A. N. Alquennah, and H. Vahedi, "Review on single-DC-source multilevel inverters: Voltage balancing and control techniques," *IEEE Open J. Ind. Electron. Soc.*, vol. 3, pp. 711–732, 2022.

[3] H. Vahedi, P.-A. Labbé, and K. Al-Haddad, "Sensor-less five-level packed U-cell (PUC5) inverter operating in stand-alone and grid-connected modes," *IEEE Trans. Ind. Informat.*, vol. 12, no. 1, pp. 361–370, Feb. 2016.

[4] H. Vahedi, K. Al-Haddad, Y. Ounejjar, and K. Addoweesh, "Crossover switches cell (CSC): A new multilevel inverter topology with maximum voltage levels and minimum DC sources," in *Proc. IECON 39th Annu. Conf. IEEE Ind. Electron. Soc.*, Vienna, Austria, 2013, pp. 54–59.

[5] H. Makhamreh, M. Trabelsi, O. Kükrer, and H. Abu-Rub, "An effective sliding mode control design for a grid-connected PUC7 multilevel inverter," *IEEE Trans. Ind. Electron.*, vol. 67, no. 5, pp. 3717–3725, May 2020.

[6] H. Makhamreh, M. Trabelsi, O. Kukrer, and H. Abu-Rub, "A simple sliding mode controller for PUC7 grid-connected inverter using a look-up table," in *Proc. IECON 45th Annu. Conf. IEEE Ind. Electron. Soc.*, 2019, vol. 1, pp. 3325–3330.

[7] I. Harbiet al., "Model-predictive control of multilevel inverters: Challenges, recent advances, and trends," *IEEE Trans. Power Electron.*, vol. 38, no. 9, pp. 10845–10868, Sep. 2023.

[8] P. Zhaoet al.., "An online neural network approximator-based model-free predictive control approach for power converters," *IEEE Trans. Power Electron.*, vol. 40, no. 10, pp. 15757–15767, Oct. 2025.

[9] Y. Fujimoto, A. Kaneko, Y. Iino, H. Ishii, and Y. Hayashi, "Challenges in smartizing operational management of functionally-smart inverters for distributed energy resources: A review on machine learning aspects," *Energies*, vol. 16, no. 3, 2023, Art. no. 1330.

[10] A. N. Alquennah, M. Chida, T. Zamzam, and M. Trabelsi, "Reinforcement learning based controller for grid-connected PUC PV inverter," in *Proc. IECON 49th Annu. Conf. IEEE Ind. Electron. Soc.*, 2023, pp. 1–6.

[11] A. N. Alquennah, M. Chida, T. Zamzam, M. Trabelsi, A. Ghrayeb, and S. Khatri, "Double deep Q networks reinforcement learning based controller for grid-connected 7-level PUC inverter," in *Proc. 4th Int. Conf. Smart Grid Renewable Energy*, 2024, pp. 1–6.

[12] A. Kermansaraviet al.., "Reinforcement learning based control of grid-connected PUC5 inverter," in *Proc. IECON 50th Annu. Conf. IEEE Ind. Electron. Soc.*, 2024, pp. 1–6.

[13] A. N. Alquennah, A. Krama, H. Abu-Rub, A. Ghrayeb, and S. Bayhan, "Model free reinforcement learning based controller for grid-tied 9-level packed-e-cell multi-level inverter," in *Proc. IEEE Energy Convers. Congr. Expo.*, 2024, pp. 1–7.

[14] A. Rajamallaiah, S. P. K. Karri, M. L. Alghaythi, and M. S. Alshammari, "Deep reinforcement learning based control of a grid connected inverter with LCL-filter for renewable solar applications," *IEEE Access*, vol. 12, pp. 22278–22295, 2024.

[15] R. kumar Mahto, P. Mishra, and A. Alam, "Design of controller for three-phase grid-connected inverter using reinforcement learning," in *Proc. 1st Int. Conf. Innov. Sustain. Technol. Energy, Mechatron., Smart Syst.*, 2024, pp. 1–6.

[16] D. Weber, M. Schenke, and O. Wallscheid, "Steady-state error compensation for reinforcement learning-based control of power electronic systems," *IEEE Access*, vol. 11, pp. 76524–76536, 2023.

[17] O. Menéndez, D. López-Caiza, L. Tarisciotti, F. Ruiz, F. Auat-Cheein, and J. Rodrıǵuez, "Assessment of deep reinforcement learning algorithms for three-phase inverter control," in *Proc. IEEE 8th Southern Power Electron. Conf. 17th Braz. Power Electron. Conf.*, 2023, pp. 1–8.

[18] A. N. Alquennah, M. A. Saleh, H. Abu-Rub, A. Ghrayeb, S. Bayhan, and M. Trabelsi, "Double deep Q networks reinforcement learning-based dynamic weighting factor of FCS-MPC for multilevel inverters," in *Proc. IEEE 18th Int. Conf. Compat., Power Electron. Power Eng.*, 2024, pp. 1–7.

[19] A. N. Alquennah, M. Trabelsi, A. Krama, H. Vahedi, and M. Mohamed-Seghir, "ANN based auto-tuned optimized FCS-MPC for grid-connected CSC inverter," in *Proc. 3rd Int. Conf. Smart Grid Renewable Energy*, 2022, pp. 1–6.

[20] M. Gheisarnejad, A. Fathollahi, M. Sharifzadeh, E. Laurendeau, and K. Al-Haddad, "Data-driven switching control technique based on deep reinforcement learning for packed e-cell as smart EV charger," *IEEE Trans. Transport. Electrific.*, vol. 11, no. 1, pp. 3194–3203, Feb. 2025.

[21] Y. Wan, Q. Xu, and T. Dragičević, "Reinforcement learning-based predictive control for power electronic converters," *IEEE Trans. Ind. Electron.*, vol. 72, no. 5, pp. 5353–5364, May 2025.

[22] Y. Tanget al.., "Deep reinforcement learning aided variable-frequency triple-phase-shift control for dual-active-bridge converter," *IEEE Trans. Ind. Electron.*, vol. 70, no. 10, pp. 10506–10515, Oct. 2023.

[23] M. Mohamed-Seghir, A. Krama, S. S. Refaat, M. Trabelsi, and H. Abu-Rub, "Artificial intelligence-based weighting factor autotuning for model predictive control of grid-tied packed U-cell inverter," *Energies*, vol. 13, no. 12, 2020, Art. no. 3107.

[24] T. Dragičević and M. Novak, "Weighting factor design in model predictive control of power electronic converters: An artificial neural network approach," *IEEE Trans. Ind. Electron.*, vol. 66, no. 11, pp. 8870–8880, Nov. 2019.

[25] R. S. Sutton, *Reinforcement Learning: An Introduction* (A Bradford Book Series). Cambridge, MA, USA: MIT Press, 2018.

[26] J. Eschmann, "Reward function design in reinforcement learning," in *Reinforcement Learning Algorithms: Analysis and Applications*. Cham, Switzerland: Springer, 2021, pp. 25–33.

[27] V. Mahajan, P. Agarwal, and H. O. Gupta, "An artificial intelligence based controller for multilevel harmonic filter," *Int. J. Elect. Power Energy Syst.*, vol. 58, pp. 170–180, 2014.

[28] A. Bakeer, I. S. Mohamed, P. B. Malidarreh, I. Hattabi, and L. Liu, "An artificial neural network-based model predictive control for three-phase flying capacitor multilevel inverter," *IEEE Access*, vol. 10, pp. 70305–70316, 2022.

[29] M. Aliet al.., "Robust ANN-based control of modified PUC-5 inverter for solar PV applications," *IEEE Trans. Ind. Appl.*, vol. 57, no. 4, pp. 3863–3876, Jul./Aug. 2021.

[30] M. Babaie, F. Sebaaly, M. Sharifzadeh, H. Y. Kanaan, and K. Al-Haddad, "Design of an artificial neural network control based on levenberg-marquart algorithm for grid-connected packed U-cell inverter," in *Proc. IEEE Int. Conf. Ind. Technol.*, 2019, pp. 1202–1207.

[31] H. Makhamreh, M. Trabelsi, O. Kükrer, and H. Abu-Rub, "Model predictive control for a PUC5 based dual output active rectifier," in *Proc. IEEE 13th Int. Conf. Compat., Power Electron. Power Eng.*, 2019, pp. 1–6.

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

[33] A. Rajamallaiahet al., "Deep reinforcement learning for power converter control: A comprehensive review of applications and challenges," *IEEE Open J. Power Electron.*, vol. 6, pp. 1769–1802, 2025.

[34] J. Yeet al.., "An overview of reinforcement learning for power electronic converters: Topology derivation, parameter design, and control implementation," *Renewable Sustain. Energy Rev.*, vol. 228, 2026, Art. no. 116591.

[35] M. Trabelsi, S. Bayhan, K. A. Ghazi, H. Abu-Rub, and L. Ben-Brahim, "Finite-control-set model predictive control for grid-connected packed-U-cells multilevel inverter," *IEEE Trans. Ind. Electron.*, vol. 63, no. 11, pp. 7286–7295, Nov. 2016.

[36] M. Trabelsi, H. Vahedi, and H. Abu-Rub, "Review on single-DC-source multilevel inverters: Topologies, challenges, industrial applications, and recommendations," *IEEE Open J. Ind. Electron. Soc.*, vol. 2, pp. 112–127, 2021.

[37] A. U. Rehman, Z. Ullah, H. S. Qazi, H. M. Hasanien, and H. M. Khalid, "Reinforcement learning-driven proximal policy optimization-based voltage control for PV and WT integrated power system," *Renewable Energy*, vol. 227, 2024, Art. no. 120590.

[38] C. Liu, Y. Wang, Q. Cui, and B. Pal, "Bi-objective reinforcement learning for PV array dynamic reconfiguration under moving clouds," *Electric Power Syst. Res.*, vol. 245, 2025, Art. no. 111579.

[39] A. Kokkinis and K. Siozios, "Fast resource estimation of FPGA-based MLP accelerators for TinyML applications," *Electronics*, vol. 14, 2025, Art. no. 247.

[40] E. E. Khodaet al., "Ultra-low latency recurrent neural network inference on FPGAs for physics applications with hls4ml," *Mach. Learn.: Sci. Technol.*, vol. 4, no. 2, 2023, Art. no. 025004.

[41] Z. Jianget al.., "Low latency transformer inference on FPGAs for physics applications with hls4ml," *J. Instrum.*, vol. 20, no. 04, 2025, Art. no. P04014.

[42] A. A. Teodoro, O. S. Gomes, M. Saadi, B. A. Silva, R. L. Rosa, and D. Z. Rodrıǵuez, "An FPGA-based performance evaluation of artificial neural network architecture algorithm for IoT," *Wireless Pers. Commun.*, vol. 127, no. 2, pp. 1085–1116, 2022.