Reinforcement Learning for Optimizing Battery Operation in Electric Vehicle Charging Microgrids

Valentin BURGAUD, Grégoire LE GOFF, Pauline KERGUS, Maurice FADEL
LAPLACE, Université de Toulouse, CNRS, INPT, UPS, Toulouse, France

ABSTRACT – This paper focuses on a data-driven control strategy for the optimal operation of a microgrid that integrates charging stations for electric vehicles (EV). To mitigate environmental impact, modern EV charging networks increasingly rely on renewable energy sources such as solar and wind power, complemented by battery storage systems. However, the intermittent nature of renewable energy presents challenges in ensuring a stable and efficient energy supply. The objective of this work is to minimize the reliance on grid electricity by optimizing battery use. A two-step approach is proposed : first, an offline optimization using linear programming (LP) identifies an ideal solution under deterministic conditions. Second, a real-time control solution is developed using reinforcement learning (RL), where a Deep Deterministic Policy Gradient (DDPG) agent is trained offline based on the optimal solutions derived from LP. The contribution of this study is a detailed exploration of the agent's tuning process, including hyperparameter selection and training strategies to improve learning efficiency. Moreover, the convergence of the trained agent is examined to ensure that the control policy remains robust to uncertainties in renewable energy generation and fluctuating consumption patterns. The simulation results validate the effectiveness of the proposed approach, demonstrating its ability to solve the problem online while ensuring stable microgrid operation across various scenarios.

Keywords – Microgrid, optimization, reinforcement learning, energy management.

1. INTRODUCTION

The widespread integration of renewable energy sources (RES) and the rapid development of electric vehicle infrastructures have revolutionized the energy landscape. Microgrids, which combine local energy production and consumption, offer a promising solution to efficiently manage energy distribution while promoting sustainability. However, the intermittent nature of RES, such as wind and solar energy, and the highly variable demand from EV charging stations pose significant challenges for optimal energy control and management [1]. These fluctuations lead to uncertainties in power availability and demand, necessitating advanced control strategies to ensure reliable and cost-effective energy distribution.

Determining optimal energy dispatch in such systems requires addressing a highly dynamic and stochastic environment. Traditional model-based approaches struggle to provide real-time adaptability in the face of rapid changes in renewable generation and EV charging patterns. This work contributes to the design of data-driven control algorithms that operate without prior system modeling, addressing the critical need for real-time energy management in microgrids with intermittent renewable energy sources and unpredictable EV charging demands.

Linear programming methods, for example, are widely used [2] to compute the optimal energy dispatch by solving a deterministic optimization problem. LP provides an optimal solution under known conditions and serves as a reference for decision making. However, its reliance on a predefined model and a perfect forecast over the considered horizon limits its applicability in real-time operations.

To overcome these limitations, reinforcement learning [3] emerges as a promising alternative, capable of handling uncertain environments in a continuous manner. Specifically, the potential of the Deep Deterministic Policy Gradient algorithm [4, 5] is explored for real-time energy control in microgrids. Unlike other RL control techniques, DDPG operates in continuous action spaces, making it well-suited for microgrid applications where battery storage control and energy dispatch decisions require fine-grained adjustments. An initial strategy consists in training the DDPG model to replicate the optimal control actions derived from the LP formulation, which serves as a reference for optimal decision making.

However, the implementation of DDPG presents several challenges. As a model-free deep reinforcement learning (DRL) algorithm, its performance is highly dependent on the choice of hyperparameters, training data, network architecture, and design of reward functions [6]. Moreover, DDPG is known to suffer from training convergence issues, such as overestimation bias and poor sample efficiency, which can hinder the convergence of the learning algorithm and lead to suboptimal or unstable control policies if not properly addressed. This study systematically investigates the impact of hyperparameter tuning and stabilization techniques on improving the robustness of the learned policy.

This study presents a comparison between the offline LP-based approach and the control policy obtained through DDPG. Our objectives are threefold :

- Optimal solution via LP : Establish a baseline for energy dispatch in a microgrid by solving the deterministic and omniscient problem using LP. This baseline provides an optimal policy under ideal conditions and serves as a reference to evaluate the RL-based approach.
- Real-Time control with DDPG : Train an DDPG agent on the optimal solutions derived from LP, enabling real-time decision-making in a dynamic environment. Given the continuous nature of energy control decisions, the DDPG is particularly suited for this task, as it allows for smooth and fine-tuned adjustments to battery operations.
- Performance Evaluation and Convergence Analysis : The DDPG policy is compared against the LP baseline in terms of efficiency and adaptability. In addition to standard performance metrics, the analysis focuses on the convergence of the learning process, addressing known challenges such as overfitting, hyperparameter sensitivity, and the exploration–exploitation trade-off.

2. METHODOLOGY AND PROBLEM RESOLUTION

2.1. *System Description and Data Collection*

The microgrid under study is specifically designed for EV charging applications, integrating multiple renewable energy sources together with a battery storage system to improve energy management and reduce reliance on the external grid. The infrastructure comprises four ABB Terra HP 150 kW fast chargers, a photovoltaic system with a peak capacity of 800 kW, and a wind turbine capable of delivering up to 900 kW. Furthermore, a 400 kWh battery storage unit is implemented, with a charge and discharge capacity of $\pm 20\ kW$, to provide additional flexibility in regulating the distribution of power within the microgrid Figure 1. Although this storage capacity is undersized relative to current technological standards and typical microgrid requirements, no precise sizing optimization was conducted in this study, as the focus is placed on control strategy evaluation rather than system design.

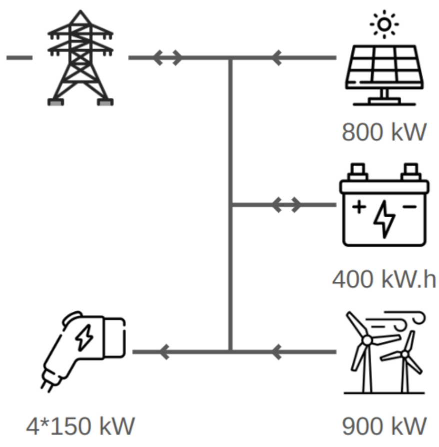

FIG. 1. System representation with power transfer.

To accurately model the system's dynamics, an aggregated EV traffic profile is constructed based on real-world data. This profile is derived from traffic sensor data collected on the A84 motorway between 2019 and 2023 [7], which provides statistics on vehicle count per hour. Heavy vehicles such as trucks are filtered to ensure that the dataset includes only passenger cars. The statistical assumption is made that the proportion of EV on the road is approximately equal to the EV registration rate in France. Applying this percentage (approximately 3.5%) to the total vehicle count, an estimated EV traffic vector is obtained.

The available traffic data set provides vehicle counts per hour. To refine the temporal resolution to match microgrid control requirements, the hourly EV counts are randomly distributed within each hour. This process generates a minute-level EV arrival profile, which is more suitable for modeling charging demand fluctuations and real-time energy management.

Renewable energy production is modeled using meteorological data sourced from a European database [8], covering the same region and timeframe as traffic data. This data set includes hourly measurements of key meteorological variables such as ambient temperature, wind speed (measured 10 meters above ground level), and solar irradiance (in W/m^2).

All these data are gathered together to create a unified dataset over a four-year period. Additionally, linear interpolation is applied between hourly points to approximate values at a one-minute time step. This ensures consistency with the EV data-traffic model and allows for more precise simulation of energy production dynamics.

Equation (1) that stands for the power balance of the grid shall be satisfied at all time :

$$\mathcal{P}_{chargers} = \mathcal{P}_{battery} + \mathcal{P}_{wind} + \mathcal{P}_{solar} + \mathcal{P}_{grid} \tag{1}$$

where $\mathcal{P}_i$ represents the power contributions of the chargers, battery storage system, wind turbine, photovoltaic panels, and the external grid.

2.2. *Offline Optimal Control Using Linear Programming*

To minimize grid dependency, the energy management problem is formulated as a LP, as shown in Equation (2). The objective function is defined as the total cost in order to optimally manage renewable power flows while dynamically responding to variable demand for EV charging. This objective leads to the formulation of the following LP optimization problem (where the problem size is 1439×4, with 1439 time steps of 1 minute and 4 variables in the LP) :

$$\min \sum_t \mathcal{P}_{grid}(t) \times c(t) \quad \text{s.t.}$$
$$\begin{cases} \mathcal{P}_{chargers} = \mathcal{P}_{battery} + \mathcal{P}_{wind} + \mathcal{P}_{solar} + \mathcal{P}_{grid} \\ SOC(t+1) = SOC(t) + \mathcal{P}_{bat}(t+1) \times T_s \\ SOC(t_0) = SOC_{init} \\ SOC(t_{d+1}) = SOC(t_d) \\ \mathcal{P}^{min}_{battery} \le \mathcal{P}_{battery}(t) \le \mathcal{P}^{max}_{battery} \\ 0.2 \times SOC_{max} \le SOC(t) \le SOC_{max} \end{cases} \tag{2}$$

In Equation (2), $\mathcal{P}_{grid}$ denotes the power drawn from the main grid and c denotes the financial time-dependent cost, SOC is the state of charge of the battery, and $\mathcal{P}^{min}_{battery}/\mathcal{P}^{max}_{battery}$ corresponds to the battery's maximum charge and discharge rates, respectively, set at $\pm 20\ kW$. These constraints enforce energy balance at each time step, govern the battery's dynamic behavior, and ensure that all physical limitations of the system are respected. The control problem is discretized with a sampling interval $T_s = 1$ minute, enabling high-resolution control actions responsive to the fast-changing conditions of renewable energy generation and electric vehicle demand. The condition $SOC(t_0) = SOC_{init}$ ensures the battery starts each day at a predefined state of charge (typically 50% of capacity), providing flexibility to absorb uncertainties. The constraint $SOC(t_{d+1}) = SOC(t_d)$ enforces daily consistency, requiring the battery to end the day at its initial charge level ; its impact on system performance is analyzed later.

While this offline LP-based approach provides an optimal energy dispatch strategy under ideal conditions (i.e., no uncertainties) that is considered as a benchmark for further investigations of partically implementable algorithms, its practical implementation in real-time settings requires accurate forecasts for renewable generation and load demand. DDPG handles uncertainty implicitly by training on a wide range of scenarios, allowing it to learn policies that generalize across variable conditions without relying on deterministic forecasts or predefined uncertainty models.

2.3. *DDPG Agent for Online Control*

RL is a machine learning approach where an agent learns to make decisions by interacting with an environment to maximize cumulative rewards. The agent observes the state $s(t)$, takes an action $\alpha(t) = \mathcal{P}_{bat}(t)$ based on its policy, and receives feedback in the form of a new state $s(t+1)$ and a reward $r(t+1)$.

This feedback loop allows the agent to improve its policy over time, as shown in Figure 3. In reinforcement learning, it is important to distinguish between the *training phase*, where the agent learns from interaction with the environment (Figure 3), and the *exploitation phase*, where the learned policy is applied to make decisions (Figure 9). Deep Reinforcement Learning (DRL), which integrates Deep Learning (DL) with RL, leverages Deep Neural Networks (DNNs) to model the agent's decision-making policy, enabling effective solutions in complex environments. One popular DRL algorithm is DDPG, which combines value-based and policy-based methods through an actor-critic approach [3]. In this section, we focus specifically on the learning process.

Principle [4, 9] is represented on figure 2 : it consists of an actor, defined by two DNNs π and π', and a critic defined by two DNNs Q and Q'. During training, the weights of these DNNs (noted as θ^{π}, $\theta^{\pi'}$, θ^{Q} and $\theta^{Q'}$) are optimized based on Equation (5) following Algorithm 1, in order to maximize the cumulative reward and the following critic loss function :

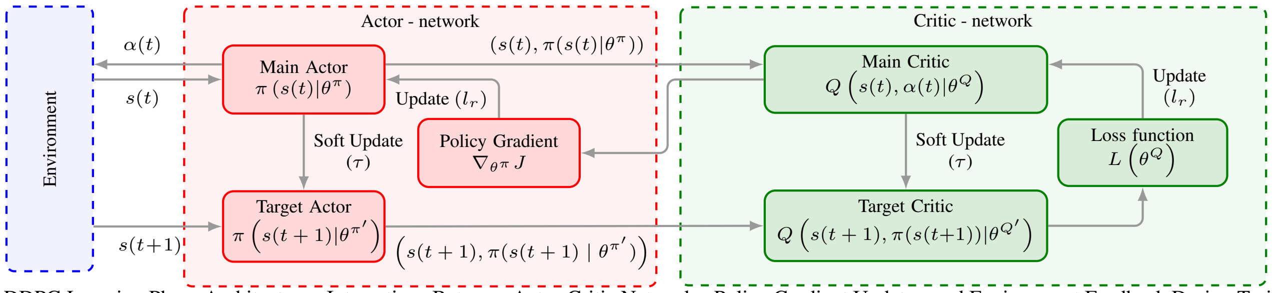
Environment
$\alpha(t)$
$s(t)$
$s(t+1)$
Actor - network
Main Actor $\pi\left(s(t)|\theta^{\pi}\right)$
Update (l_r)
Soft Update (τ)
Policy Gradient $\nabla_{\theta^{\pi}} J$
Target Actor $\pi\left(s(t+1)|\theta^{\pi'}\right)$
$(s(t), \pi(s(t)|\theta^{\pi}))$
$\left(s(t+1), \pi(s(t+1) \mid \theta^{\pi'}\right))$
Critic - network
Main Critic $Q\left(s(t), \alpha(t)|\theta^{Q}\right)$
Soft Update (τ)
Update (l_r)
Loss function $L\left(\theta^{Q}\right)$
Target Critic $Q\left(s(t+1), \pi(s(t+1))|\theta^{Q'}\right)$

FIG. 2. DDPG Learning Phase Architecture : Interactions Between Actor-Critic Networks, Policy Gradient Updates, and Environment Feedback During Training.

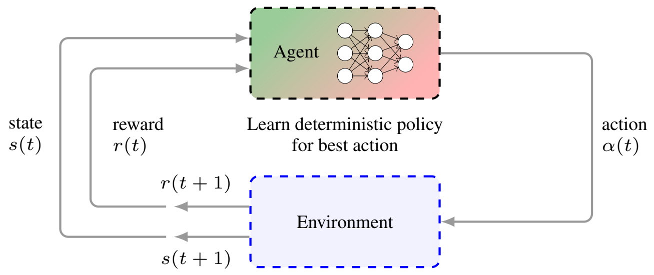

FIG. 3. The agent-environment interaction during training in RL.

$$L(\theta^{Q}) = \underset{(s(t),\alpha(t),r(t),s(t+1))\sim\mathcal{D}}{\mathbb{E}}\left[\left(Q(s(t),\alpha(t)|\theta^{Q}) - y(t)\right)^2\right]$$
$$y(t) = r(t) + \lambda Q\left(s(t+1), \pi(s(t+1)|\theta^{\pi'})|\theta^{\pi'}\right) \tag{3}$$

In this equation, $L(\theta^{Q})$ denotes the loss function minimized by the DDPG algorithm, where θ^{Q} represents the parameters of the critic network. The critic approximates the action-value function $Q(s(t), \alpha(t)|\theta^{Q})$, which estimates the expected cumulative reward obtained by taking action $\alpha(t)$ in state $s(t)$ and then following the current policy thereafter. The immediate reward received after executing action $\alpha(t)$ is denoted by $r(t)$, and λ is the discount factor that weights future rewards to balance short-term and long-term gains.

The target value function $y(t)$ is composed of the immediate reward $r(t)$ plus the discounted estimate of the next state's value, which is computed by the target actor network $\pi(s(t+1) \mid \theta^{\pi'})$. The parameters $\theta^{\pi'}$ of this target actor network are a delayed copy of the main actor network parameters θ^{π}, a design that helps to avoid the computationally expensive and often infeasible explicit maximization over continuous action spaces. By minimizing the mean squared error between the predicted Q-value and this target, the algorithm effectively minimizes the temporal difference error, a fundamental concept in reinforcement learning that drives value function improvement.

The update of the main actor parameters θ^{π} is performed using the deterministic policy gradient [4], which aims to maximize the expected Q-value estimated by the critic. This update is expressed as :

$$\nabla_{\theta^{\pi}} J \approx \mathbb{E}_{s(t)\sim\mathcal{D}}\left[\nabla_{\alpha(t)} Q\big(s(t), \alpha(t)|\theta^{Q}\big) \cdot \nabla_{\theta^{\pi}} \pi\big(s(t)|\theta^{\pi}\big)\right] \tag{4}$$

Here, the gradient is computed in two steps : first, the derivative of the critic Q with respect to the action $\alpha(t)$, evaluated at $\alpha(t) = \pi(s(t)|\theta^{\pi})$, and second, the derivative of the policy π with respect to its parameters θ^{π}. This gradient guides the update of the actor to select actions that maximize the critic's estimated value.

The actor-critic architecture of DDPG thus involves two interacting networks : the actor network θ^{π}, which is trained to produce actions that maximize the critic's value estimates, and the critic network θ^{Q}, which learns to evaluate the quality of those actions accurately. The use of target networks alongside exploration strategies ensures that the training process remains stable and converges reliably over time. Key hyperparameters influencing this process include the learning rate (l_r), which controls the step size for gradient updates and balances stability with speed of learning ; the soft update rate (τ), which governs the gradual update of target networks using the rule :

$$\theta' \leftarrow \tau\theta + (1-\tau)\theta' \tag{5}$$

It promotes smooth convergence ; and the exploration noise parameter (σ), which injects noise into the actions to encourage exploration early in training and decays over time to favor exploitation of the learned policy.

This overall learning process can be succinctly summarized in the following simplified learning steps :

Algorithm 1: DDPG Learning Steps (Simplified)

```
  Input: Current state $s(t)$
  Output: Updated actor and critic networks $\theta^{\pi}$ and $\theta^{Q}$
1 Select action $\alpha(t)$ using actor $\pi(s(t))$;
2 Execute action $\alpha(t)$, observe next state $s(t+1)$;
3 Evaluate $(s(t), \alpha(t))$ with main critic $Q$;
4 Predict next action using target actor $\pi(s(t+1))$;
5 Estimate next value using target critic
    $Q(s(t+1), \pi(s(t+1)))$;
6 Compute loss $L(\theta^{Q})$ between critic and target value;
7 Update critic network to minimize loss;
8 Update actor network using policy gradient $\nabla_{\theta^{\pi}} J$;
9 Soft-update target networks towards main networks;
```

Despite its advantages, DDPG is sensitive to hyperparameters and may experience instability of the learning process. Inaccurate Q-value estimates can lead to divergence, while excessive training can cause overfitting, reducing the model's ability to generalize. These issues are addressed in subsequent sections.

2.4. *DDPG Agent Training*

To achieve real-time battery control in a microgrid, a DDPG agent is implemented in Python using PyTorch with CUDA (*Compute Unified Device Architecture*) acceleration. The problem is discretized with a one-minute time step, aligning with the dataset resolution. Each step represents a transition where the agent observes the state, takes an action, and receives a reward.

The environment representation includes temporal, meteorological, and operational features to guide decision-making. The observation space consists of normalized time variables (month, hour, and minute) transformed using sine and cosine functions to preserve periodicity. The day of the month is scaled to $[0, 1]$. Meteorological data includes solar irradiance and wind speed, normalized to their maximum observed values. Operational indicators include a binary variable for vehicle presence, the queue size normalized to a predefined maximum, the battery state of charge scaled by its capacity, and battery power normalized by the maximum discharge rate. Predictive elements like the number of expected vehicles and forecasted cumulative solar and wind energy are also included, scaled for numerical convergence.

All variables are normalized within predefined ranges to ensure efficient learning and prevent disparities in feature magnitudes. The action space consists of a single continuous variable representing battery power, normalized between -1 and 1, and scaled according to the maximum charge and discharge rates. The DDPG agent learns to regulate power flow, optimizing for long-term energy balance and vehicle service reliability through the following reward function :

$$r(t) = -\text{penalty}(t) \times (\mathcal{P}_{battery}(t) - \mathcal{P}^{\text{ref}}_{battery}(t))^2 \quad (6)$$

The reward function minimizes the squared deviation between the selected battery power $\mathcal{P}_{battery}(t)$ and the reference $\mathcal{P}^{\text{ref}}_{battery}(t)$, with the penalty term dynamically increasing in response to violations of system constraints such as SoC bounds or abrupt power variations.

2.5. *DDPG Hyperparameter Optimization and Convergence Analysis*

The performance and convergence of DDPG algorithms are highly dependent on hyperparameter selection. Poorly chosen values can cause instability, vanishing gradients, or convergence to sub-optimal policies [10]. To mitigate this, we use Optuna, a hyperparameter optimization framework that explores candidate values within predefined bounds. Each Optuna trial trains a complete DDPG agent using one set of hyperparameters and evaluates its performance based on the reward function's value and trajectory. The optimization process combines Bayesian search and early stopping criteria to ensure both efficiency and robustness.

The final set of optimal parameters, applied in all experiments, is summarized in Table 1 :

TABLEAU 1. Hyperparameters for the DDPG Agent

Parameter	**Value**
Discount Factor (γ)	0.96
Actor Learning Rate (a_{lr})	1×10^{-4}
Critic Learning Rate (c_{lr})	6×10^{-4}
Network Update Speed (τ)	1×10^{-3}
Batch Size (batch_size)	128
Hidden Layer Width 1 ($l1_{width}$)	1024
Hidden Layer Width 2 ($l2_{width}$)	512
Hidden Layer Width 3 ($l3_{width}$)	256
Exploration Noise Multiplier (noise)	0.5
Ornstein-Uhlenbeck Mean Reversion (μ)	0.996
Mean Reversion Rate (θ)	0.01
Noise Magnitude (σ)	0.05
Replay Buffer Size	128

These hyperparameters were selected after over 50 Optuna trials, each using different random seeds to reduce variance. To enhance exploration in continuous action spaces, we follow [4] by adding Ornstein-Uhlenbeck (OU) noise, which introduces perturbations following :

$$dX_t = \beta(\mu - X_t)dt + \sigma dW_t, \quad (7)$$

where X_t is the noise state, μ the mean, β the mean reversion rate, σ the volatility, and dW_t a Wiener process.

Gradient loss analysis, especially in the actor network, is key to understanding the convergence of policy learning [11]. While the critic network's gradients primarily inform value estimation, the actor's gradients—derived from the policy gradient—directly refine the decision-making policy. Monitoring these gradients helps understand the learning dynamics : vanishing gradients indicate stagnation or overly conservative updates, while large gradients may suggest instability or divergence.

For a well-tuned DDPG agent converging to an optimal policy, actor gradients stabilize after an initial increase during early training. If the agent converges to a suboptimal policy, gradient magnitudes may remain high, indicating persistent updates that fail to improve performance, possibly due to improper exploration, poor policy quality, or inaccurate value function approximation.

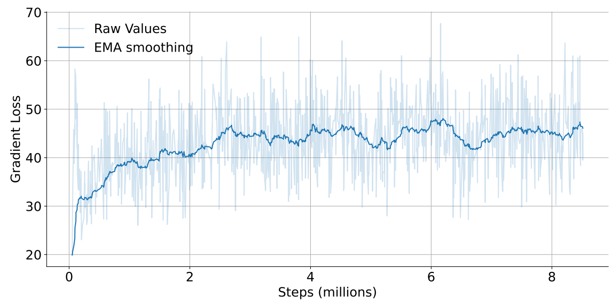

FIG. 4. Gradient magnitudes in actor networks during training.

Figure 4 shows the evolution of gradient magnitudes in both actor and critic networks. Initially, gradients increase as learning begins. Notably, the actor's gradient magnitude rises and stabilizes around 40. This plateau suggests convergence to a suboptimal policy, with continuous updates that do not significantly improve performance. To better visualize the gradient evolution and reduce noise in the results, Exponential Moving Average (EMA) smoothing is applied, which helps highlight long-term trends. The EMA is defined as :

$$\hat{g}_t = \eta g_t + (1 - \eta)\hat{g}_{t-1} \quad (8)$$

where $\hat{g}_t$ is the smoothed gradient at time step t, g_t is the raw gradient, and $\eta = 0.04$ is the smoothing factor. This smoothing process helps filtering short-term fluctuations and provides a clearer view of the underlying learning dynamics.

Analyzing the weight distribution in the first layer of both networks provides further insight into DDPG convergence. Weight evolution, especially in early layers, reveals issues such as network saturation, vanishing/exploding activations, and overfitting. The first layer shapes feature representations, and its weight statistics can indicate stable learning. In well-behaved training, weight means should remain within a bounded range, avoiding excessive drift or collapse. Excessive drift signals instability or overfitting, while near-zero means suggest underfitting or ineffective learning due to conservative updates.

For the actor network, the first layer's mean weights determine how input states are transformed into latent representations guiding action selection. Stable weight distributions support consistent policy refinement. In the critic network, the mean weights corresponding to both state and action inputs in the Q-function are analyzed. Balanced weight means ensure proper consideration of state and action features, crucial for accurate policy evaluation and improvement.

The following figures illustrate the evolution of the mean weights in the first layer of the actor Figure 5, the critic's action input path 6, and the critic's state input path 7 throughout training. As shown, all three weight means stabilize within moderate ranges after an initial period of adaptation, indicating convergence towards a stable representation space and supporting the observed convergence behavior of gradient magnitudes.

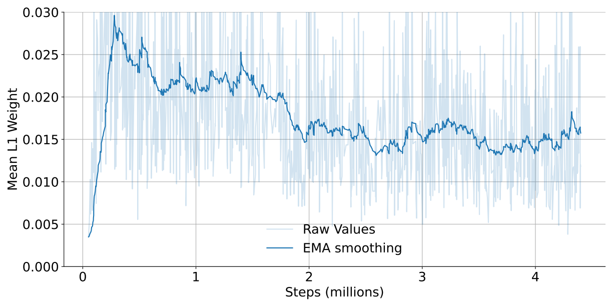

FIG. 5. Mean weights (θ^{π}) in the first layer of the actor network during training.

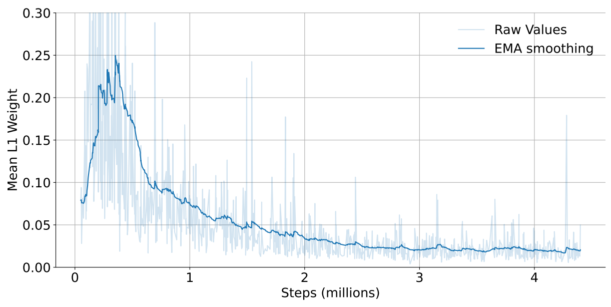

FIG. 6. Evolution of mean weights (θ^{Q}) in the first layer of the critic network related to **action inputs** during training.

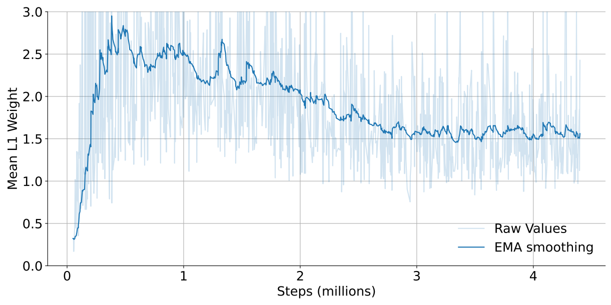

FIG. 7. Evolution of mean weights (θ^{Q}) in the first layer of the critic network related to **state inputs** during training.

To assess the performance and learning progression of the DDPG agent, the evolution of cumulative returns during training is monitored. The agent was trained on a dataset, with training samples randomly selected to cover the entire dataset.

Following this initialization, the training was initiated with parameter updates performed at each environment step (one-minute resolution). The evaluation of policy performance was performed every two days in simulated time. The evaluation protocol, defined in the function below, computes the cumulative reward (return) over three different randomly selected days from the dataset. For each evaluation, the agent performs deterministic actions and the average return across the three days is logged :

The figure below illustrates the progression of cumulative returns over 4.5 million time steps. This corresponds to evaluating the full dataset three times, i.e., $(1439 \times 365 \times 3) \times 3$ steps, ensuring that each day in the dataset could be explored multiple times to reinforce learning diversity and generalization.

Algorithm 2: Policy Evaluation

```
1 Initialize total_score ← 0;
2 for $i \leftarrow 1$ to 3 do
3     Select a random day;
4     Run the agent on that day and accumulate rewards
        into total_score;
5 return total_score/3;
```

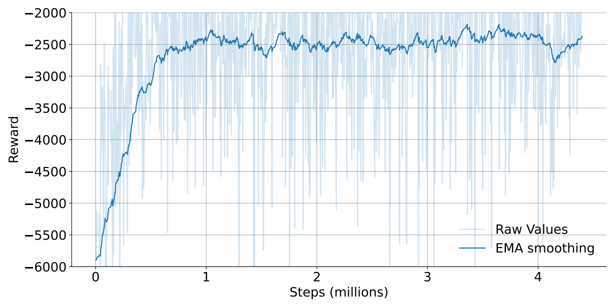

FIG. 8. Evolution of cumulative reward during training.

As shown in Figure 8, the DDPG agent exhibits steady improvement in cumulative return, with convergence towards a sub-optimal policy observed approximately one-third into the total training steps. Beyond this point, the return oscillates around a relatively stable value without reaching the theoretical optimum (zero cumulative penalty), indicating convergence to a policy that is locally, but not globally, optimal. This observation aligns with the actor loss trends and weight mean analyses discussed earlier, reinforcing the conclusion that the agent stabilizes around a consistent, albeit sub-optimal, decision-making strategy. The combined analysis of reward evolution, gradient behavior, and weight statistics offers a comprehensive understanding of the agent's learning dynamics and convergence properties.

3. ANALYSIS OF RESULTS

We now consider the exploitation phase, where the agent interacts with the environment using a previously trained neural network with the optimal hyperparameters identified in the previous section. This trained policy aims to select the best actions to control battery power in real time, maximizing renewable energy utilization and minimizing grid dependence.

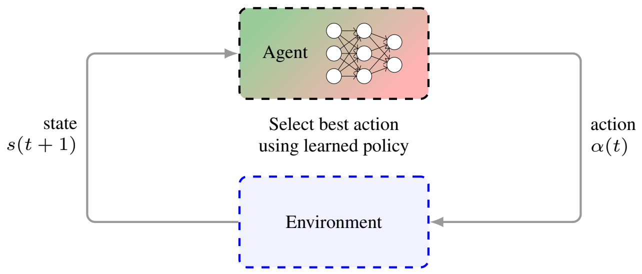

FIG. 9. The agent-environment interaction during exploitation in RL.

Figure 10 shows a comparative analysis of the battery power $\mathcal{P}_{battery}$ and the state of charge (SOC) over a 24-hour period for both the LP and DDPG solutions. The DDPG agent's behavior closely mirrors that of the LP, demonstrating its ability to approximate the optimal control policy in real-time scenarios. However, despite this success, a critical limitation is evident in the DDPG agent's performance : it learned a suboptimal policy, as shown in Figure 8. The agent fails to fully utilize the battery capacity. This limitation arises from the agent's inability to fully capture the dynamics of the reward function (6), which does not adequately emphasize efficient battery usage and lacks a physical sense, making it difficult to interpret in terms of real-world energy management. As a result, while the agent minimizes errors between predicted and actual battery control actions, it does not fully exploit the battery's discharge capabilities, leading to a suboptimal control policy.

Further examination of the worst day (where the agent achieves the worst cumulative reward over a day) in Figure 11 shows significant deviations in the DDPG agent's predictions compared to the optimal LP solution. In such cases, the agent fails to adhere to the LP solution, highlighting the need for further refinement in policy learning and exploration.

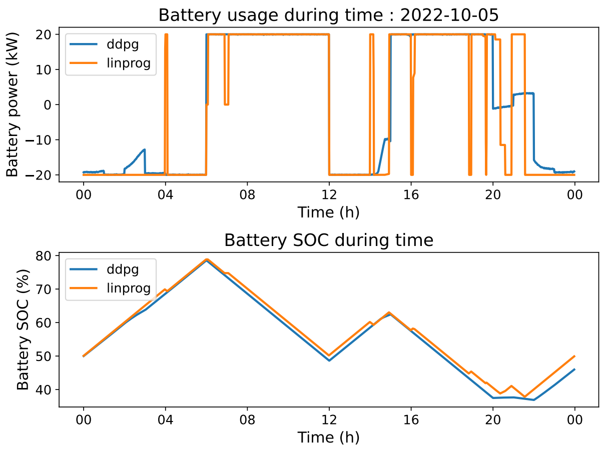

FIG. 10. Evaluation dataset day (05/10/22) with cumulative reward (Figure 8) closest to zero (-1350).

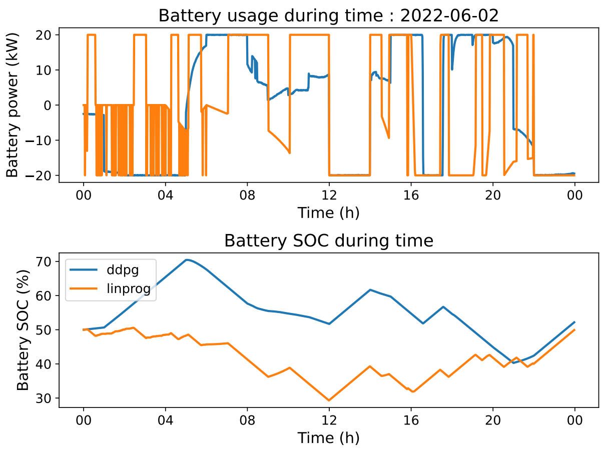

FIG. 11. Evaluation dataset day (02/06/22) with worst cumulative reward (Figure 8) (-5753).

4. CONCLUSIONS

This study demonstrates the potential of the DDPG algorithm as a viable solution for real-time battery control in microgrids. The agent shows promising results in approximating optimal LP policies but also reveals areas for improvement, particularly in constraint handling and exploration efficiency. Its failure to fully utilize the battery's discharge rate and maintain the SOC within optimal bounds are critical limitations that need to be addressed.

Future work should focus on refining the reward function to better capture environment dynamics, including efficient battery usage and SOC limits. By employing an automatic hyperparameter optimization method rather than relying solely on commonly used values from the literature, a different set of hyperparameters was identified, leading to a better-performing agent. Additionally, exploring alternative continuous reinforcement learning algorithms, such as Soft Actor-Critic (SAC), could help mitigate some of DDPG's limitations by enabling better exploration and more robust policy learning.

Evaluating the performance of the DDPG agent under fault and disturbance scenarios is also essential to assess its resilience in real-world microgrid management. An online learning setup with injected disturbances could allow the agent to adapt to unforeseen changes in renewable generation and consumption, enhancing its applicability in dynamic environments.

In conclusion, while the DDPG algorithm shows potential for real-time battery control in microgrids, refinements are needed to fully exploit its capabilities. Future research will explore alternative RL algorithms such as SAC and evaluate their performance relative to MPC. A comprehensive comparison between RL and MPC will clarify trade-offs between RL and MPC, using Key Performance Indicators (KPIs) such as energy cost minimization, constraint satisfaction, and robustness. This work lays the foundation for advanced control systems that combine both approaches to enhance microgrid resilience and efficiency.

5. REFERENCES

[1] F. Ahsan et al, "Data-driven next-generation smart grid towards sustainable energy evolution : techniques and technology review" in Protection and Control of Modern Power Systems, vol. 8, no. 3, pp. 1-42, Jul. 2023, doi : 10.1186/s41601-023-00319-5.

[2] A. Almomani, M. S. Alam and S. Ali Arefifar, "Review of Power System Resilience : Operation Stages, Vulnerabilities, and Modeling Approaches" in IEEE International Conference on Electro Information Technology (eIT), Eau Claire, WI, USA, 2024, pp. 297-302, 2024, doi : 10.1109/eIT60633.2024.10609944.

[3] E. H. Sumiea, S. J. Abdulkadir, H. S. Alhussian, S. M. Al-Selwi, A. Alqushaibi, M. G. Ragab, and S. M. Fati, "Deep deterministic policy gradient algorithm : A systematic review" in Heliyon, vol. 10, no. 9, 2024, doi : 10.1016/j.heliyon.2024.e30697.

[4] T. P. Lillicrap, J. J. Hunt, A. Pritzel, et al., "Continuous control with deep reinforcement learning" in Proceedings of the 4th International Conference on Learning Representations (ICLR), 2019. doi : 10.48550/arXiv.1509.02971.

[5] T. Zhang, et al, "A Bayesian Deep Reinforcement Learning-Based Resilient Control for Multi-Energy Micro-Grid" in IEEE Transactions on Power Systems, vol. 38, no. 6, pp. 5057-5072, Nov. 2023, doi : 10.1109/TPWRS.2023.3233992.

[6] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing Function Approximation Error in Actor-Critic Methods" in *CoRR*, vol. abs/1802.09477, 2018, doi : 10.48550/arXiv.1802.09477

[7] Traffic data. Cerema. https://www.cerema.fr/fr/mots-cles/donnees-trafic

[8] Photovoltaic Geographical Information System. PVGIS. https://joint-research-centre.ec.europa.eu/photovoltaic-geographical-information-system-pvgis_en

[9] S. Silver, G. Lever, N. Heess, T. Degris, D. W. L. L. M. Riedmiller, and W. Zaremba, "Deterministic Policy Gradient Algorithms" in Proceedings of the 31st International Conference on Machine Learning (ICML), Beijing, China, 2014, pp. 387-395, 2014.

[10] G. Matheron, N. Perrin, and O. Sigaud, "Understanding Failures of Deterministic Actor-Critic with Continuous Action Spaces and Sparse Rewards" in *Proc. Artificial Neural Networks and Machine Learning – ICANN 2020*, Cham, Switzerland : Springer, 2020, pp. 308–320, doi : 10.1007/978-3-030-61616-8_25.

[11] T. Zhang, et al, "A Bayesian Deep Reinforcement Learning-Based Resilient Control for Multi-Energy Micro-Grid" in IEEE Transactions on Power Systems, vol. 38, no. 6, pp. 5057-5072, Nov. 2023, doi : 10.1109/TPWRS.2023.3233992.