scientific reports

OPEN

Enhanced twin delayed DDPG with prioritized experience replay and Noisy Nets for regional economic dispatch

Chang Xu[1,2]✉, Naoki Hayashi[2], Masahiro Inuiguchi[2], Wong Jee Keen Raymond[1], Hazlie Mokhlis[1,3] & Hazlee Azil Illias[1]

Integrating renewable energy into power systems introduces significant challenges in balancing generation costs and grid stability, necessitating advanced solutions for the Economic Dispatch Problem (EDP). While classical mathematical and meta-heuristic methods face scalability and computational efficiency limitations, reinforcement learning (RL) offers a promising alternative due to its adaptability to high-dimensional and dynamic environments. This study employs Twin Delayed DDPG (TD3), an enhanced version of Deep Deterministic Policy Gradient (DDPG). TD3 integrates Prioritized Experience Replay (PER) and Noisy Networks (Noisy Nets) for the EDP in a regional microgrid with photovoltaic (PV) generation. PER improves sample efficiency by prioritizing high-error transitions, while Noisy Nets enhance exploration through adaptive parameter noise. Experiments demonstrate that combining these techniques with TD3 achieves a 54.6% reduction in testing operation cost and a 95.3% decrease in cumulative power unbalance compared to the baseline TD3. The improvements are validated across various deterministic and stochastic RL models, with TD3+PER+Noisy Nets outperforming others in cost efficiency and stability. The findings demonstrate the proposed approach's capability to optimize microgrid dispatch while providing a scalable and practical framework for power system control.

The widespread adoption of sensors has enabled the development of smart grids that offer grid operators detailed insights into power generation and transmission. At the same time, higher penetration of renewable energy on the generation side introduces new challenges. Output from renewable sources is intermittent and variable because it depends on natural conditions. A primary operational objective in smart-grid energy management is to dispatch fuel-based generators efficiently to maximize renewable-energy utilization. Consequently, these issues are commonly framed as the EDP in power systems.

Solutions to the EDP can be categorized into three main approaches: classical mathematical methods, meta-heuristic techniques, and AI-based algorithms[1,2]. Classical mathematical methods are the earliest approaches developed for solving EDPs. These methods are generally applied to simplified problems, where the objective focuses on generation costs, typically modeled by quadratic functions, and the system features relatively few constraints[1]. A representative classical mathematical method based on quadratic programming, which considers line flow and emission constraints, was proposed in 1998 and tested on a power system with five generation units and ten buses[3]. However, classical mathematical methods generally perform poorly when applied to large-scale and complex power systems. Significant system simplifications are often required to ensure computational feasibility[4], particularly when the system involves many constraints[5]. As modern power systems continue to grow in complexity, the computational time required by classical mathematical methods increases exponentially, rendering them unsuitable for addressing contemporary large-scale optimization problems.

In contrast, meta-heuristic techniques[6,7] offer a promising alternative for optimizing modern complex power systems[8]. These methods provide approximate yet acceptable solutions within reasonable computational time and are applicable across various optimization problems. Among them, population-based meta-heuristics excel at global exploration and at escaping local optima. The trade-off is higher computational cost and additional parameter tuning[9]. For solving EDPs, trajectory-based meta-heuristics have been explored in some studies[10], but

[1]UM Power and Energy System Research Group, Department of Electrical Engineering, Faculty of Engineering, Universiti Malaya, 50603 Kuala Lumpur, Malaysia. [2]Graduate School of Engineering Science, The University of Osaka, Toyonaka 560-8531, Japan. [3]Photonics Research Centre, Universiti Malaya, 50603 Kuala Lumpur, Malaysia. ✉email: u385272b@ecs.osaka-u.ac.jp

population-based meta-heuristics are more widely adopted[1]. In particular, numerous hybrid algorithms have been proposed to address various dispatch-related challenges. Awadallah et al.[11] introduced the Memetic Salp Swarm Algorithm (MSSA), which integrates the global exploration capability of the Salp Swarm Algorithm with an adaptive local search strategy to enhance solution accuracy in EDPs. Shaban et al.[12] proposed the Growth Optimizer (GO), a novel metaheuristic inspired by human learning and introspection processes, which demonstrated superior robustness and outperformed various contemporary algorithms across multiple benchmark dispatch systems. Similarly, Fahim et al.[13] developed a hybrid Jaya and Teaching–Learning-Based Optimization (TLBO) algorithm that integrates complementary mechanisms from different metaheuristics to handle non-smooth cost functions and complex operational constraints, thereby improving solution reliability.

Although meta-heuristic methods outperform classical mathematical approaches, the increasing scale of modern power systems leads to a substantial expansion of the search space, resulting in prolonged computational times required to obtain satisfactory solutions[14]. This limitation becomes even more pronounced when the dispatch horizon is divided into finer time intervals, as the number of decision variables grows proportionally with the time step resolution. Similarly, larger system scales, with more distributed generators, storage units, and network constraints, expand the search space. In addition, meta-heuristic methods often require extensive parameter tuning and multiple iterations to prevent premature convergence, which increases the computational burden and limits their practicality for real-time or near-real-time applications.

With advancements in deep learning, RL-based approaches[15,16] have been introduced into power system control[17,18]. Compared to meta-heuristic methods, RL-based approaches are more capable of handling high-dimensional data and are well-suited for online execution due to their rapid decision-making ability based on the current state[19]. Chiş et al. [20] developed an RL-based strategy for electric vehicle (EV) charging. Linear programming (LP) was first employed to determine the optimal daily charging actions under various scenarios based on historical datasets, and the Fitted Q-Iteration algorithm[21] was subsequently applied to learn the optimal policy. This approach effectively reduced users' long-term EV charging costs while simplifying daily charging decisions by eliminating the need to solve complex linear programming problems each day. Hou et al. [22] proposed a hybrid approach combining mixed-integer linear programming (MIP) and deep Q-learning (DQN). After training the DQN model, online execution was enhanced using commercial MIP solvers to ensure that all operational constraints were strictly enforced, which standard DQN policies alone could not guarantee. Additionally, Claessens et al. [23] integrated a convolutional neural network (CNN) into the Fitted Q-Iteration framework to approximate the Q-function, thereby extracting state-time features from thermostatically controlled loads and reducing electricity costs while achieving near-optimal performance in simulations. A summary of the above-reviewed EDP research works is presented in Table 1.

Beyond the scope of economic dispatch, reinforcement learning has also been combined with graph-based and federated frameworks to address challenges in distribution systems. For instance, an unrolled spatiotemporal graph convolutional network (USGCN) was applied to state estimation and forecasting by capturing renewable-related spatiotemporal correlations [24]. A full-model-free adaptive graph deep deterministic policy gradient (FAG-DDPG) approach was proposed for multi-terminal soft open point voltage control, leveraging graph attention to enhance feature extraction and policy learning [25]. Furthermore, a prototype federated reinforcement learning framework with a physics-aware spatiotemporal transformer (STT-PFRL) was introduced for privacy-preserving and robust voltage regulation under renewable uncertainties [26]. These studies demonstrate the broader applicability of reinforcement learning and graph-based techniques beyond dispatch problems.

In RL-based power system control, many studies have focused on integrating RL with other algorithms to improve performance rather than enhancing the RL algorithm structure itself. However, RL algorithms can be structurally modified to improve performance in control applications. To incorporate RL improvement techniques into power system control, this study proposes a TD3 controller enhanced with PER and Noisy Nets to address the EDP problem in regional power grids. PER enhances the sample efficiency of the standard TD3 model by prioritizing transitions that contribute more significantly to learning, thereby accelerating convergence and improving overall performance. In parallel, Noisy Nets introduce adaptive exploration by

Category	Model	Key features	Limitations	Reference
Classical mathematical methods	Quadratic programming	Suitable for small-scale systems; handles generation cost with polynomial approximation	Poor scalability; not suitable for complex modern systems	3
Meta-heuristic techniques	Memetic salp swarm algorithm (MSSA)	Combines global exploration of SSA with adaptive local search; improves convergence and solution accuracy	Additional computational effort due to hybridization; requires parameter tuning	11
	Growth optimizer (GO)	Inspired by human learning and introspection; robust performance across multi-scale EDP benchmarks	Algorithm still new; requires further validation in diverse scenarios	12
	Jaya and teaching–learning-based optimization (TLBO)	Integrates complementary mechanisms from different metaheuristics; effective for non-smooth cost functions and operational constraints	Complexity of coordination may increase implementation difficulty	13
AI-based / RL methods	Fitted Q-Iteration + LP	Offline training via LP; online decision-making for EV charging	Requires historical data; less generalizable to unseen scenarios	20
	DQN + MIP	Combines DQN policy learning with strict operational constraint enforcement via MIP	Online execution depends on commercial solvers; more computationally intensive	22
	CNN + Fitted Q-Iteration	Learns state-time features for control of thermostatically controlled loads	Training requires large data; tailored to specific applications	23

Table 1. Summary of representative methods for solving economic dispatch problems (EDPs).

injecting parameterized noise into the network, enabling the agent to explore the action space more effectively and avoid premature convergence to suboptimal policies. The primary contributions of this study are as follows:

1. A hybrid reinforcement learning controller is developed by integrating PER and Noisy Nets within TD3. This structural enhancement improves sample efficiency and adaptive exploration without relying on external hybridization frameworks or optimization solvers, simplifying deployment.
2. The enhancement modules (PER and Noisy Nets) are modular and transferable across different reinforcement-learning controllers, including DDPG, SAC, and MPO, demonstrating generalizability across deterministic and stochastic policy models in power system control.
3. In a regional power grid environment with realistic pricing, renewable generation, and battery storage, the proposed controller outperforms multiple baselines in operational cost minimization and power balance, indicating suitability for near real-time dispatch.

The remainder of this paper is structured as follows: Section II introduces the background of improvement techniques and control models, including DDPG, SAC, and MPO. Section III presents the structure of the proposed PER+Noisy Nets TD3 controller. Section IV describes the modeling of the regional grid used as an environment for the RL controller and details the proposed model's hyperparameters. Section V discusses numerical results and comparative analysis. Finally, Section VI provides the conclusions.

RL improvement techniques and controlling models

This section introduces several well-known RL improvement techniques and widely adopted RL-based control models. Some of these methods are also selected for comparison with the proposed approach in subsequent sections.

With the advancement of deep learning, RL has gained significant attention, with documented cases of exceeding human baselines on specific tasks. Unlike supervised learning, which requires manually labeled datasets and is limited to performing at most as well as the training data, RL enables models to achieve superhuman performance[27]. Hessel et al.[28] tested six different DQN variants on 57 Atari 2600 games, with three of them (DQN+PER, Dueling DQN, and Distributional DQN) achieving higher scores than human experts. Beyond classic video games, RL has also demonstrated superior performance in more complex scenarios. Vinyals et al.[29] employed a multi-agent RL algorithm with an actor-critic structure to play StarCraft II, achieving a ranking above 99.8% of officially ranked human players after reaching the highest Grandmaster level. These results underscore the strong control performance and stability of RL algorithms.

RL improvement techniques

To enhance RL performance, numerous improvement techniques have been proposed. Some of the most notable include Double Q-learning, PER, Dueling Network Architecture, Multi-step Learning, Distributional RL, and Noisy Nets. Compared to the standard DQN architecture, these techniques improved performance in Atari benchmarks. When these enhancements were integrated into a single agent, Rainbow[28], it achieved the highest scores, exceeding human performance by more than a factor of two. This demonstrates that combining multiple improvements can substantially boost model performance.

The problems addressed in these game-related studies are highly complex. The RL agent must process state representations composed of pixel-based images, which are influenced by game mechanics and randomness. Furthermore, the action space in such environments is often high-dimensional, requiring the agent to select from a vast range of possible actions[30]. In contrast, RL-based controllers for power systems operate in a significantly less complex environment. Given this difference in complexity, applying these RL improvement techniques to power system control problems should be feasible and often sufficient for power-system control problems.

Mathematical formulation of the economic dispatch problem

In the environment adopted in this study, generator fuel costs are represented as convex quadratic functions, while grid transactions are modeled linearly with sales valued at a discounted coefficient. Penalties on power unbalance beyond the grid exchange capacity are modeled linearly, and basic operational constraints such as generator ramping and battery state-of-charge updates are also included.

However, several features reflecting physical operating limits make the formulation of the EDP problem non-convex. Distributed generators can either be off or operate only within a bounded range, the battery dynamics involve clipping at the minimum and maximum SOC, and the grid exchange is asymmetric due to capped transactions, different buy and sell prices, and penalties for excess or shortage. These characteristics introduce discontinuities and non-smooth dynamics.

As a result, the environment represents the EDP as a non-convex optimization problem with quadratic costs and piecewise-linear constraints, rather than as a simple convex quadratic program. This complexity motivates the use of reinforcement learning, which can learn effective dispatch policies offline and then apply them in real time without repeatedly solving a difficult optimization problem.

RL-based controlling models

DQNs[31] introduced neural networks to approximate Q-values in reinforcement learning, enabling effective control in discrete action spaces. However, DQNs face limitations in handling continuous action problems.

To address this, the actor-critic framework[32] has become a foundational architecture in RL. In this framework, the actor learns a policy $\pi_\theta(s)$ to select actions, while the critic estimates the value function $Q(s, a)$ to evaluate and guide the actor's updates. Traditional actor-critic methods adopt on-policy learning with stochastic policies, where the actor outputs a probability distribution over actions. Based on the type of policy used, actor-critic models are typically categorized into deterministic and stochastic approaches.

Deterministic policy models
Deep Deterministic Policy Gradient (DDPG)[33] is a widely used RL algorithm designed for continuous action spaces. As its name suggests, DDPG employs a deterministic policy, meaning the actor directly outputs a specific action without sampling. To mitigate training instability in actor-critic architectures, DDPG introduces target networks to stabilize the critic updates. Additionally, it adopts an experience replay buffer to enable off-policy learning, improving data efficiency and reducing training costs. However, DDPG suffers from an overestimation bias in Q-values. Since the critic updates by maximizing the Q-value, network approximation errors or noise can lead to overestimated Q-values. To address this issue, TD3 was proposed as an improvement over DDPG.

Stochastic policy models
SAC[34] is a stochastic policy algorithm based on the maximum entropy framework. Unlike conventional RL algorithms that solely aim to maximize the expected cumulative reward, SAC seeks to optimize both reward maximization and policy entropy simultaneously. The standard RL objective is to learn a policy that maximizes the expected cumulative reward:

$$\pi^* = \arg\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right]. \tag{1}$$

In contrast, SAC optimizes for an augmented objective function that includes an entropy regularization term:

$$\pi^* = \arg\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{\infty} \left(\gamma^t r(s_t, a_t) + \alpha H\left(\pi(\cdot|s_t)\right)\right)\right], \tag{2}$$

where the entropy term $H\left(\pi(\cdot|s_t)\right)$ is defined as

$$H\left(\pi(\cdot|s_t)\right) = -\mathbb{E}_{a\sim\pi}\left[\log \pi(a|s_t)\right]. \tag{3}$$

In the above formulations, π represents the policy and π^* denotes the optimal policy. The expectation operator $\mathbb{E}[\cdot]$ ensures the optimization focuses on the long-term average reward rather than any individual high-reward instance. γ is the discount factor, which balances short-term and long-term rewards. $r(s_t, a_t)$ represents the reward function. The entropy term $H\left(\pi(\cdot|s_t)\right)$ measures the action variety of the policy, with a larger entropy indicating greater exploration. The temperature coefficient α is a tunable hyperparameter that controls the importance of the entropy term in the objective function. A larger α results in more stochastic policies, whereas a smaller α leads to more deterministic behavior. During training, α is automatically adjusted to balance exploration and exploitation.

Compared to TD3, SAC differs primarily in its actor structure. While TD3 employs a deterministic policy network that directly outputs actions, SAC utilizes a stochastic policy network that outputs an action distribution, from which actions are sampled using the reparameterization trick. Despite this difference, SAC and TD3 share a common feature in their critic design. Similar to TD3, SAC employs two independent critic networks to mitigate the problem of Q-value overestimation.

By incorporating the entropy term into the objective function, SAC enhances exploration without relying on external noise to introduce action variability. This leads to a more diverse set of actions, thereby improving overall performance. Additionally, this modification enhances the model's robustness, as the policy is less likely to become overly dependent on specific actions.

MPO[35] is another stochastic policy algorithm inspired by Expectation-Maximization (EM) techniques and probabilistic inference. Its key feature is constraining policy updates to remain close to the previous policy, ensuring training stability. MPO decomposes policy optimization into two alternating steps:

1. E-Step: Given state *s*, sample *N* actions a_i using the current policy, then compute Q-values $Q(s, a_i)$ using the critic. The probabilities of these actions are then adjusted using a SoftMax transformation:

$$w(a_i|s) = \frac{\exp\left(\frac{Q(s,a_i)}{\eta}\right)}{\sum_{j=1}^{N} \exp\left(\frac{Q(s,a_j)}{\eta}\right)}, \tag{4}$$

where the temperature parameter η controls the update intensity. A higher η produces smoother updates, while a lower η leads to more aggressive updates.

2. M-Step: The policy is updated via weighted maximum likelihood to favor high-Q actions.

By constraining updates through temperature regulation, MPO ensures stable training. However, while this temperature-based regulation offers improved stability, it also introduces trade-offs. The constraint on policy updates results in increased computational complexity. This may limit the practical applicability of MPO in environments with high-dimensional action spaces. Moreover, the reduced exploration due to temperature constraints makes the algorithm more susceptible to getting stuck at a local minimum. In such scenarios, the policy might converge prematurely to suboptimal solutions. This limits the model's ability to explore a diverse set of actions and learn a more globally optimal policy.

For EDP in power systems, the complexity is significantly lower than in high-dimensional RL applications. Moreover, real-world constraints limit the available data for training. Using excessively complex control models may not necessarily yield better performance. It could lead to unstable or inefficient training. Considering the complexity of the environment, this study proposes an improved TD3 controller for optimizing the operation of a regional power grid. The previously discussed DDPG, SAC, and MPO algorithms are employed as benchmarks to evaluate and compare the performance of the proposed improved TD3 model.

Research methodology

In this study, the combined improvement techniques are introduced into the power system control model to enhance its performance when addressing the EDP. Two improvement techniques are selected for integration, namely PER and Noisy Nets. The overall framework of the proposed TD3+PER+Noisy Nets method is illustrated in Fig. 1. The following subsections provide detailed explanations of each component and the training process.

TD3

TD3[36] is an enhancement over DDPG and forms the core of the controller design in this study. Compared to DDPG, TD3 introduces several key improvements:

- *Dual Critic Networks* TD3 maintains two critic networks simultaneously and computes two Q-values for each state-action pair. During the update process, the smaller Q-value is selected to mitigate the overestimation bias that is commonly observed in single-critic architectures. This conservative approach helps ensure that the estimated Q-values are not overly optimistic.
- *Delayed Policy Updates* The actor network is updated less frequently than the critic networks. This strategy guarantees that the critic networks can converge to more accurate Q-value estimates before the actor is updated, thereby promoting training stability.
- *Target Policy Smoothing* To further stabilize the target Q-value estimation, target policy smoothing is employed. When calculating the target Q-value, noise is added to the action generated by the target policy network. Specifically, the perturbed action is computed as follows:

$$a' = \pi_{\theta'}(s') + n, \quad n \sim \text{clip}\left(N(0,\sigma), -c, c\right), \tag{5}$$

where $\pi_{\theta'}(s')$ denotes the action produced by the target policy network for the next state s', and n is noise sampled from a Gaussian distribution with mean 0 and standard deviation σ, which is then clipped to the interval $[-c, c]$. Importantly, this noise is not introduced for the purpose of exploration; rather, it encourages the critic to consider Q-values over a local neighborhood of actions, thereby promoting a smoother Q function that is less prone to overestimation.

The selection of TD3 as the primary controller in this study is motivated by its demonstrated stability and robustness in continuous control tasks. The additional structural improvements in TD3 (especially in stabilizing

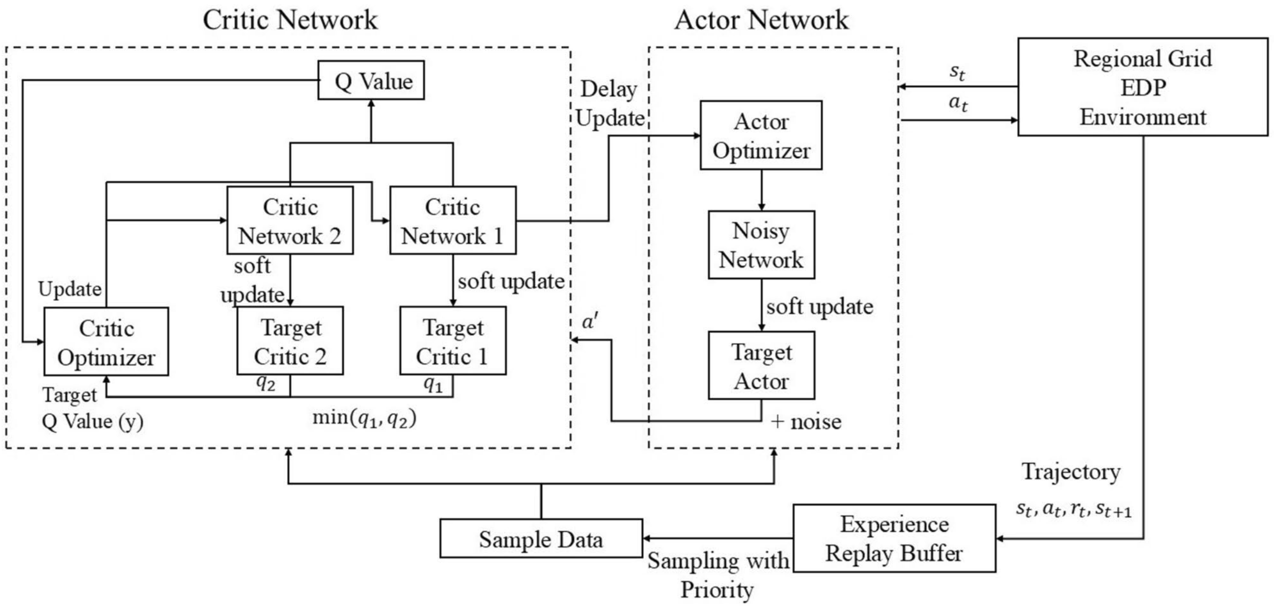

Fig. 1. The structure of the proposed TD3+PER+Noisy Nets controller.

the critic network) are critical when combining the controller with other techniques. In particular, the subsequent integration of PER and Noisy Nets necessitates a robust baseline to ensure that a reliable policy can be obtained.

PER

Prioritized Experience Replay (PER)[37] is an extension of the conventional Experience Replay mechanism. In traditional Experience Replay, a replay buffer is used to store transitions, i.e., the state, action, reward, and next state (s, a, r, s') obtained from interactions with the environment. This allows the RL algorithm to perform off-policy updates by sampling experiences from the buffer, thereby increasing sample efficiency by reusing past experiences multiple times without requiring additional environment interactions. Typically, experiences in the buffer are sampled uniformly with the probability:

$$P(i) = \frac{1}{N}, \tag{6}$$

where *N* is the capacity of the replay buffer. In contrast, PER assigns a priority to each transition based on its temporal-difference (TD) error. The sampling probability for a given transition *i* is defined as:

$$P(i) = \frac{p_i^a}{\sum_j p_j^a}, \tag{7}$$

where $p_i = |\delta_i| + \varepsilon$, δ_i denotes the TD error of transition *i*, ε is a small positive constant that ensures every transition has a non-zero probability of being sampled (even when $\delta_i = 0$), and *a* is a hyperparameter that determines the degree of prioritization. When $a = 0$, the sampling becomes uniform, larger values of *a* bias the sampling process towards transitions with larger TD errors.

Since non-uniform sampling introduces bias (i.e., transitions with high TD errors are sampled more frequently, potentially reducing sample diversity), importance sampling weights are incorporated into the loss function to counteract this bias. The importance sampling weight for a transition *i* is given by:

$$w_i = \left(\frac{1}{N \cdot P(i)} \right)^{\beta}, \tag{8}$$

where β is another hyperparameter that adjusts the extent of the correction. The weighted loss function is then expressed as:

$$L = \frac{1}{m} \sum_{i=1}^{m} w_i \left(y_i - Q(s_i, a_i) \right)^2, \tag{9}$$

with *m* being the minibatch size, y_i the target value for transition *i*, and $Q(s_i, a_i)$ the current Q-value estimate. PER effectively accelerates learning by focusing on transitions that are more informative for the learning process. This process is particularly advantageous in settings with high interaction costs. A typical example is a power-system environment, where operating costs and power-unbalance metrics must be computed repeatedly. Furthermore, PER is generalizable to any off-policy algorithm utilizing a replay buffer, and its code implementation is relatively simple.

Noisy Nets

Enhancing exploration in RL is essential for improving the quality of the solutions obtained. A common strategy is to inject external noise into the policy, for instance, by adding Gaussian noise directly to the deterministic action output:

$$a = \pi(s) + n, \quad n \sim N(0, \sigma^2), \tag{10}$$

where *a* means the output action and *n* denotes the added noise, and $\pi(s)$ is the original action determined by the policy based on state *s*. While this method is straightforward and can facilitate exploration, the fixed noise intensity may be insufficient in complex environments. To overcome this limitation, Noisy Nets[38] are introduced as a more adaptive approach. As can be seen from Fig. 2, instead of adding noise directly to the action output, Noisy Nets incorporate noise into the parameters (weights and biases) of the neural network itself. This approach allows the magnitude of the noise to be learnable, thereby endowing the network with a more powerful and context-sensitive exploration capability. Consider a standard linear layer represented as:

$$y = wx + b, \tag{11}$$

where *w* and *b* denote the weight matrix and bias vector, respectively, and *x* and *y* represent the input and output of the layer. When Noisy Nets are employed, the layer is modified as follows:

$$y = (\mu^w + \sigma^w \odot \varepsilon^w)\, x + \mu^b + \sigma^b \odot \varepsilon^b. \tag{12}$$

In this expression, μ^w and μ^b are the deterministic parameters, while σ^w and σ^b represent the learnable noise scaling factors. The variables ε^w and ε^b are random noise samples drawn from a predefined distribution

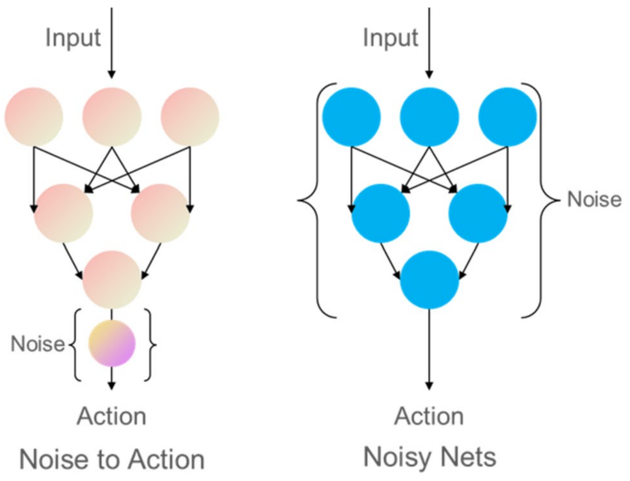

Fig. 2. The comparison between Noisy Nets and conventional noise injection.

(typically a Gaussian) and $\odot$ denotes element-wise multiplication. By integrating noise directly into the network parameters, Noisy Nets achieve adaptive exploration that is directly tied to the agent's interactions with the environment. In the proposed method, only the actor network is replaced with a Noisy Net architecture, as applying Noisy Nets to the critic networks was found to introduce training instability.

Training process

```
Require: Environment $\mathscr{E}$, stochastic policy $\pi$ with noise, replay buffer $\mathscr{B}$
Ensure: Trained actor and critic networks
 1: Initial Data Collection:
 2: for $t = 1$ to $10,000$ do
 3:     Collect transition $(s_t, a_t, r_t, s_{t+1})$ using $\pi$ with exploration noise
 4:     Store in replay buffer $\mathscr{B}$
 5: end for
 6: Off-Policy Training:
 7: for episode = 1 to 450 do
 8:     Sample minibatch from $\mathscr{B}$ with priority based on TD errors
 9:     Update critics using TD targets and priorities
10:     Update actor policy using policy gradient
11:     Soft-update target networks periodically
12:     Collect new samples using current policy and add to $\mathscr{B}$
13:     Record losses and evaluation metrics
14:     Save best model parameters when performance improves
15: end for
16: Post-Training Evaluation
```

Algorithm 1. Training Framework of this study

This study evaluates four models: TD3, DDPG, SAC, and MPO. All models are based on an actor–critic architecture, and the fundamental training framework is maintained consistently across experiments, as outlined in Algorithm 1. The training process comprises several stages:

1. *Initial Data Collection* Initially, a stochastic policy augmented with extra exploration noise is employed to interact with the environment. A total of 10,000 steps are collected and stored in the replay buffer.
2. *Off-Policy Training* Following the initial data collection, off-policy training is conducted over 450 episodes. At each training iteration, a minibatch of 256 transitions (s_t, a_t, r_t, s_{t+1}) is sampled from the PER buffer using priority sampling. The sampling probability for each transition is determined by its TD error, so that transitions with larger errors are more likely to be selected. For each sampled transition, the following update steps are performed:

a. *Critic Update* The target actor network generates a new action for the next state s_{t+1}, incorporating additional policy noise for robustness. Subsequently, two target critic networks compute the Q-values for the generated action. The smaller of the two Q-values is selected to reduce the overestimation bias and is used to calculate the TD target (also known as target Q-value). The current critic networks then compute their respective Q-values for the sampled transitions, and the TD errors are computed accordingly. These TD errors inform the update of the critics' parameters via the loss function. Additionally, the computed TD errors are used to update the priorities of the corresponding transitions in the PER buffer.
b. *Actor Update* The actor network, implemented as a Noisy Net, produces an action for the current state s_t. A single critic network then evaluates the Q-value corresponding to this action. The actor's parameters are updated in the direction that maximizes the Q-value, following a policy gradient method.
c. *Target Network Update* After a fixed number of iterations, soft updates are applied to synchronize the target networks with their corresponding online networks. This step is critical for maintaining the stability of the training process.

3. *Continuous Data Collection and Model Saving* As training progresses and the performance of the actor improves, new samples are continuously collected from interactions with the environment and added to the replay buffer. This process ensures that the buffer always contains up-to-date experiences, which is crucial for the continual refinement of the model. During each episode, the critic loss, actor loss, and various environmental evaluation metrics are recorded. The parameters of the actor and critic networks are saved when the operational cost reaches its optimal performance, thereby designating the best model for subsequent testing.
4. *Post-Training Evaluation* After the completion of training, the best model is evaluated in an independent testing environment using unseen data. Key performance metrics such as overall operational cost and power unbalance are calculated to validate the model's effectiveness in practical applications.

In summary, each component of the proposed method contributes a distinct advantage: TD3 ensures the stability of the critic during training; PER enhances sample efficiency and accelerates convergence; and Noisy Nets provide an adaptive mechanism for improved exploration. The synergistic integration of these techniques results in a high-performance controller that is robust and reliable for continuous control tasks in power system operations.

Experiment settings

This section details the modeling of the regional grid used as the environment in this study and outlines the hyperparameters for the TD3 controller and the integrated improvement techniques. The experimental setup is designed to rigorously evaluate the performance of the RL controllers under realistic operational conditions.

Modeling of regional grid

The environment employed in this study originates from a publicly available framework introduced by Shengren et al.[22]. The original study formulated the problem using a MIP solver to guarantee strict satisfaction of operational constraints. Although the initial purpose of that framework was constraint-oriented optimization, it also provides a well-structured setting for evaluating RL controllers under realistic economic dispatch conditions. Accordingly, the environment was employed in this study with an extended training period of 11 months and a testing period of 1 month.

As can be seen from Fig. 3, the environment comprises two main components: a regional grid and a main grid. The regional grid consists of a solar farm, three distributed generators with distinct generation parameters, a battery energy storage system (BESS), and various loads. The BESS is modeled by considering its energy capacity, state-of-charge (SOC) limits, maximum charging and discharging rates, round-trip efficiency, and a degradation factor. At each time step, the SOC is updated based on the charging or discharging action while constrained within the allowable range. This simplified formulation captures the essential operational characteristics of the BESS while remaining computationally efficient for integration into the reinforcement learning framework. For the solar farm and loads, simulation data and collected real-world data are used rather than detailed numerical models. In contrast, the generation cost of the fuel-based generators and the charging/discharging process of the BESS are represented using mathematical models. The main grid functions as a backup supplier. When the

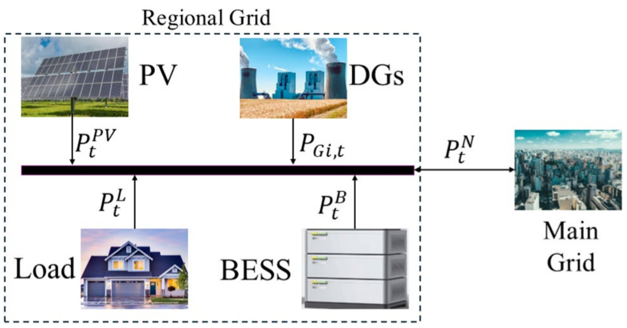

Fig. 3. The components of the environment used in the experiments.

regional grid's generation is insufficient, electricity is purchased from the main grid; conversely, if the regional grid produces excess power, it is sold back to the main grid to maintain power balance. Real-time electricity prices are defined based on historical data. In addition to operational costs, the analysis accounts for power unbalance. The transmission line linking the main grid and the regional grid has a specified power limit. When a generation shortfall requires imports beyond this limit, an unbalance penalty is incurred. This penalty is added to the total operational cost to reflect the higher expense associated with disrupted power balance.

The overall objective in this environment is to minimize the total operational cost of the regional grid. Achieving this objective requires an optimal strategy for utilizing the BESS to counteract the intermittent nature of solar power output and for efficiently scheduling power generation among the three fuel generators. The reward function for the environment is defined as follows:

$$r_t = -\frac{1}{2000}\left(C_{\text{BESS}} + \sum_{i=1}^{3} C_{G_{i,t}} + C_t^E + C_{\text{penalty}}\right), \tag{13}$$

$$SOC_{t+1} = \max\left(SOC_{\min}, \min\left(SOC_{\max}, SOC_t + \frac{\eta_c P_t^{ch} - P_t^{dis}/\eta_d}{C}\right)\right), \tag{14}$$

$$C_{G_{i,t}} = a_i\left(P_{G_{i,t}}\right)^2 + b_i P_{G_{i,t}} + c_i, \tag{15}$$

$$C_t^E = \begin{cases} p_t P_t^N, & P_t^N < 0, \\ \beta p_t P_t^N, & P_t^N \geq 0, \end{cases} \tag{16}$$

$$C_{\text{penalty}} = P_{\text{unbalance}} \cdot \kappa_{\text{penalty}}, \tag{17}$$

where C_{BESS} denotes the operational cost of the BESS (set to 0 in this experiment), $C_{G_{i,t}}$ represents the generation cost of fuel-based generator i at time t with cost coefficients a_i, b_i, and c_i, and C_t^E accounts for the revenue from electricity transactions at time t (with p_t as the real-time electricity price and β set to 0.5 to reflect the price differential between buying and selling electricity). P_t^N is the power transmitted between the main grid and the regional grid. C_{penalty} is the penalty cost associated with power unbalance, with κ_{penalty} set to 20 (substantially higher than the standard electricity price to emphasize the importance of maintaining balance). For the BESS, the parameters include the total capacity C, SOC limits $SOC_{\min}$ and $SOC_{\max}$, maximum charging and discharging power (P_t^{ch}, P_t^{dis}), and charging and discharging efficiencies (η_c, η_d).

The divisor of 2000 serves as a *reward-scaling constant* that limits the numerical magnitude of the reward signal. This value approximates the upper bound of the total daily operating cost in the studied environment and is used solely to stabilize optimization. Since the reward is defined as the negative total cost, the scaled reward becomes a negative quantity, typically around -200. The intent of this scaling is not to normalize rewards to the range [0, 1], but rather to prevent excessively large gradients and to facilitate stable and comparable learning dynamics across experiments. For further details regarding the operational constraints of the target environment, please refer to the original work [22].

The test environment used in the experiments has the same structure and parameter settings as the training environment. The key difference lies in the dataset used: the original study[22] utilizes a one-year dataset containing load demand, electricity price, and PV generation data. In the training environment, only the first 11 months of this dataset are used for model training, while the test environment is evaluated on the data from the final month. This separation ensures that the model is assessed on unseen data, providing a robust evaluation of its generalization ability and real-world applicability. The dispatch problem considered in this study corresponds to a day-ahead scheduling framework. Each episode covers a 24-hour horizon with an hourly resolution, i.e., the time interval between decision steps is set to 1 hour.

Hyperparameters selection

The hyperparameters for the RL training process and the PER mechanism are summarized in Table 2a. These settings are critical to ensuring the rapid convergence and stability of the models under investigation.

This study evaluates multiple models, including the baseline TD3, TD3 augmented with Noisy Nets, TD3 integrated with PER, and TD3 combining both PER and Noisy Nets. Additionally, the study considers DDPG, SAC, and MPO models enhanced with both PER and Noisy Nets. All models share similar architectural frameworks and hyperparameter configurations, with key parameters such as Target Step, Max Capacity, Discount Factor, and Soft Update Factor kept consistent with those reported in the original work[22]. The baseline model, a plain TD3 without any enhancement techniques, serves as the benchmark for performance comparison.

The episode numbers of 300 and 450 were selected based on preliminary experiments, which demonstrated that the models converged within this range even without the use of PER. Furthermore, incorporating PER further accelerates convergence. The choice of the number of neurons per hidden layer and the number of layers was constrained by the computational overhead introduced by Noisy Nets. Since each neuron in a Noisy Net requires individual noise generation, the network size was limited to maintain reasonable training times while preserving adequate model capacity for the environment. The learning rate, batch size, and optimizer were selected following standard practices and demonstrated sufficient effectiveness in experiments, negating the need for more complex alternatives.

The PER hyperparameters used in this experiment are summarized in Table 2b. These parameters are configured to maintain a balance between prioritizing high-error transitions and ensuring sufficient sample

Parameter	Value	Description
(a) Hyperparameters for the training process		
Episode number	300, 450	The total number of training episodes was set to 450 for TD3 and 300 for the remaining models.
Learning rate	1e-4	Learning rate used by the optimizer.
Optimizer	Adam	The Adam optimizer is selected for its robustness, eliminating the need for a more complex alternative.
Mini batch size	256	Number of samples drawn from the replay buffer per update.
Layer number	4–5	Depth of the RL network.
Neuron number per hidden layer	64	Chosen based on a trade-off between computational efficiency and model performance.
Discount factor (γ)	0.995	Balances immediate and future rewards.
Soft update factor (τ)	1e-2	Controls the smoothness of target network updates to prevent abrupt parameter changes and stabilize the policy network's output.
(b) Hyperparameters of PER		
Max capacity	50,000	Maximum size of the experience replay buffer; a larger buffer ensures that high-priority samples do not dominate the updates.
Alpha	0.5	Prioritization exponent (α) that controls the influence of the TD error on the sampling probability.
Beta start value	0.4	Initial value of the importance sampling weight (β), which corrects for the bias introduced by prioritized sampling.
Beta frames	2e6	The number of frames over which β is linearly increased from the start value to 1.
Target step	1000	Number of samples collected before updating the replay buffer.

Table 2. Hyperparameters for the training process and PER.

diversity. The values for Alpha and the Beta start value (0.5 and 0.4, respectively) are adopted from the recommended settings in the Rainbow network[28], which have been demonstrated to yield robust outcomes. During training, β is increased gradually from 0.4 to 1 over 2 million frames (as controlled by the Beta Frames parameter), thereby ensuring that the importance sampling corrections become increasingly effective and contribute to stabilizing the learning process. Preliminary tests explored alternative combinations of Alpha and initial Beta value, but none outperformed the default configuration; therefore, the recommended settings were retained.

For Noisy Nets, the sole hyperparameter is the noise parameter σ, which governs the magnitude of the noise added to the network parameters. In this study, the initial standard deviation for the noise in the Noisy Linear layers is set to 0.1. This relatively small noise magnitude was selected based on experimental observations indicating that larger noise levels can adversely affect performance. Specifically, the environment used in this study is relatively simple. It contains limited inherent randomness and involves fine-grained control of the BESS's charging and discharging states. In such an environment, large exploration noise can be detrimental because it may disrupt the precise control actions required for optimal performance.

For the DDPG model employed in this study, Gaussian exploration noise with a standard deviation of 0.5 is used. To ensure sustained exploration during training, an exploration rate parameter is implemented; initially, the exploration rate is set to 1 (ensuring that noise is always added to the action), and it gradually decreases to 0.3 by the end of training, meaning that 30% of the actions are perturbed by noise. To maintain fairness in comparison, the TD3 model also incorporates the same Gaussian noise mechanism. In contrast to DDPG, TD3 further integrates policy noise (set to 0.2) and adopts a delayed policy update schedule (with the actor updated after every two critic updates) to enhance training stability.

For the SAC model, the target entropy is the most critical hyperparameter. While a common default is $-d$ (where d is the action dimensionality), this study sets the target entropy to $-0.25 \times d$. Preliminary experiments revealed that setting a smaller target entropy (e.g., -1 or -0.5) promotes excessive exploration, which is counterproductive in power system control tasks that require precise adjustments. The temperature parameter for SAC is initially set to 1 and is updated automatically during training.

Finally, for MPO, the hyperparameters include the temperature parameter (η) and the number of action samples. To conserve computational resources, the number of action samples is set to 16, which is acceptable given the relative simplicity of the environment. The temperature parameter η is set to 0.2, representing an aggressive configuration intended to encourage the reuse of high Q-value samples to learn superior policies. However, experimental results indicate that MPO underperforms relative to algorithms with stronger exploration mechanisms. This outcome likely reflects MPO's emphasis on stable updates in high-dimensional settings rather than on extensive exploration.

Overall, the hyperparameters in this study were selected to facilitate rapid and stable convergence to near-optimal solutions. Given the complexity of the models and the integration of multiple improvement techniques, the hyperparameter space is vast, and an exhaustive grid search was impractical under the available computational resources. The experiments were implemented in Python and conducted on a system equipped with an Intel i9-10900K processor, 128 GB of RAM, and an NVIDIA RTX 3070 GPU.

Numerical analysis

This section presents a comprehensive numerical analysis of the performance of various RL controllers. Both the baseline forms and enhanced versions are analyzed, where the enhanced versions refer to RL controllers integrated with improvement techniques. The results demonstrate the efficacy of integrating RL improvement techniques into the EDP in power systems and their broad applicability to different off-policy training models.

Model	Training operation cost ($)	Training unbalance (kW)	Testing operation cost ($)	Testing unbalance (kW)	Convergence needed episodes
TD3+Both	993.69	0	9559.16	35.49	~200
TD3+PER	1075.20	1.86	13040.55	151.64	~200
TD3+Noisy	1227.61	0	17553.63	164.69	~350
TD3	1660.57	275.05	21074.13	755.32	~350

Table 3. Result comparison between TD3 controllers and its variants.

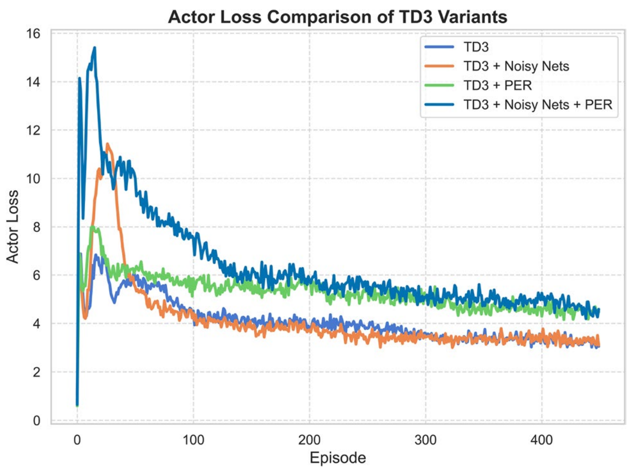

Fig. 4. Actor loss comparison of TD3 variants.

Fairness and reproducibility were ensured by employing a fixed set of 15 random seeds per model, which mitigated training stochasticity.

Comparison of TD3-based controllers

The performance of the baseline TD3 model is compared with its variants incorporating Noisy Nets, PER, or both. Table 3 summarizes the key performance metrics, including training operation cost, training unbalance, testing operation cost, testing unbalance, and the number of episodes required for convergence. The reduction of actor loss during the training process is illustrated in Fig. 4. A lower actor loss indicates that the policy network generates actions that align more closely with the critic's value estimation, thus reducing the discrepancy between predicted and optimal actions. Consequently, the reduction of actor loss reflects the progressive stabilization of the policy and serves as evidence of model convergence.

It should be noted that the Training Operation Cost and Training Unbalance are measured at the end of a series of training episodes, while the Testing Operation Cost and Testing Unbalance represent cumulative values over the entire test period. In the original study, the total dataset spans one year[22]. In this study, the training phase utilizes data from the first 11 months, while the testing phase is conducted on data from the final month. Evaluating cumulative operational costs over the full 11-month training period would be computationally prohibitive; therefore, such assessments are deferred to the testing phase to ensure computational efficiency while maintaining the integrity of performance evaluation.

The reported "Convergence Needed Episodes" is provided as an approximate range. This is due to occasional instability in the critic's training process, which can transiently affect the actor's performance, sometimes resulting in a sudden increase in the actor's loss. In some cases, particularly when PER is employed, convergence can occur as early as 80 episodes; in other instances, the behavior is comparable to that of the plain TD3 model. Overall, however, the incorporation of PER consistently accelerates convergence.

A detailed comparison of the data in Table 3 reveals that both improvement techniques, when applied individually, yield significant enhancements in performance relative to the baseline TD3 model. Specifically, both training and testing operation costs and unbalance are substantially reduced. Notably, PER is particularly effective in accelerating convergence and improving performance, an outcome that aligns with findings reported in the Rainbow literature[28]. When PER and Noisy Nets are combined in the TD3 framework, the improvements are most pronounced: the training operation cost is reduced by 40.2% and the testing operation cost by 54.6% relative to the baseline. Moreover, the training set achieves a zero unbalance, and the cumulative testing unbalance is reduced by 95.3%. In this configuration, Noisy Nets enhance the agent's exploration capabilities, while PER increases sample utilization through prioritized sampling of high temporal-difference error transitions. Their combined effect fosters an effective balance between exploration and exploitation, enabling the model to converge more rapidly and to identify near-global optima in solving the complex EDP.

As shown in Fig. 4, after experiencing an initial period of varying degrees of fluctuation, all TD3 variants eventually converge during training. Notably, the TD3+PER+Noisy Nets variant, which achieves the best overall performance, also exhibits the highest actor loss among all models. While this may seem counterintuitive at first glance, it is a reasonable outcome. A lower actor loss typically indicates that the actor's selected actions are assigned higher Q-values by the critic network. However, when PER is introduced, the critic is trained on samples with higher TD errors more frequently, making the learning target more complex and potentially less stable. As a result, the critic provides more challenging feedback to the actor, leading to an increase in actor loss. This does not imply poorer actor performance; rather, it reflects the increased difficulty of learning under a more dynamic and informative critic, which can ultimately result in a more effective control policy.

Figure 5 presents the dispatch results of three distributed generation units (DG1–DG3) and the BESS over the first three days of the testing set. It can be observed that the TD3+PER+Noisy model outperforms the plain TD3 baseline in balancing load demand and coordinating storage operations.

First, regarding the DG unit outputs, the plain TD3 model exhibits rigid or unstable scheduling patterns. For instance, DG2 and DG3 remain idle during most periods, causing the system to rely heavily on a single generator or the BESS, which increases both operational cost and unbalance. By contrast, the TD3+PER+Noisy model achieves a more flexible allocation, with DG1–DG3 alternately supplying power at different times. This results in a more balanced dispatch that reduces both cost and system unbalance.

Second, the BESS power and SOC trajectories highlight further differences. Under the plain TD3 model, charging and discharging events are limited, and SOC remains at relatively low levels, indicating underutilization of the storage capacity. In contrast, the TD3+PER+Noisy model dynamically adjusts BESS operation in response to load and PV fluctuations: charging during low-demand periods and discharging during peak demand periods. The resulting SOC follows a regular cyclical pattern, demonstrating improved peak shaving and valley filling. This behavior enhances the stabilizing role of storage and ensures reliable operation under load variability.

Overall, the dispatch results confirm that the TD3+PER+Noisy model not only achieves lower operational cost and reduced unbalance in numerical terms, but also produces more realistic and practically interpretable

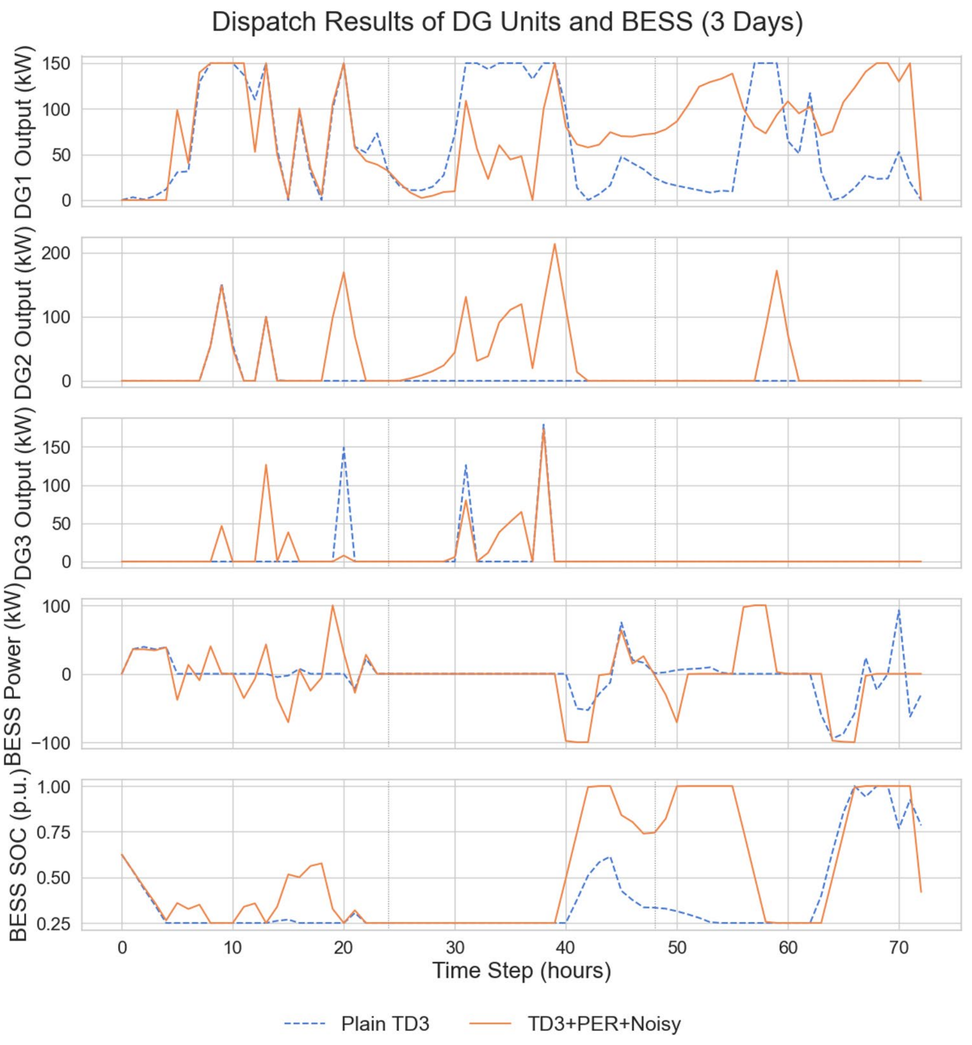

Fig. 5. Dispatch results of DG units and BESS under different models. Illustration of DG1–DG3 and BESS scheduling in the first three days of the testing set.

Model	Training operation cost ($)	Training unbalance (kW)	Testing operation cost ($)	Testing unbalance (kW)
SAC+Noisy+PER	1180.56	213.80	14370.42	339.54
SAC	1538.84	1.87	16413.88	399.66
Improvements	23.28%	–	12.45%	15.04%
DDPG+Noisy+PER	1907.14	0	20773.47	184.52
DDPG	2147.64	0	21575.91	454.90
Improvements	11.20%	–	3.72%	59.44%
MPO+Noisy+PER	3823.32	166.45	31500.51	1488.39
MPO	4056.81	116.21	38124.18	1740.82
Improvements	5.76%	–	17.37%	14.50%

Table 4. Result comparison between RL controllers.

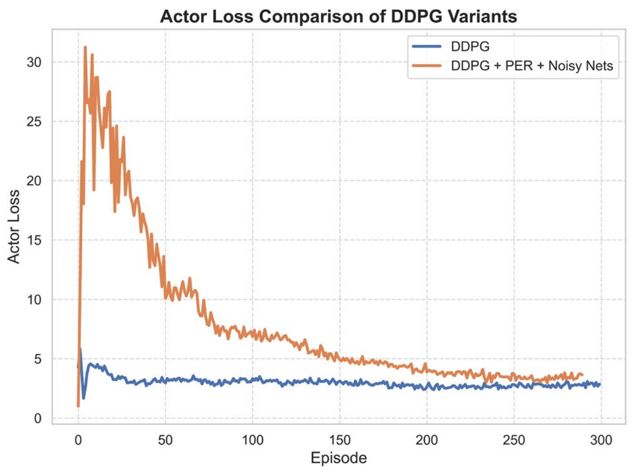

Fig. 6. Actor loss convergence of DDPG variants.

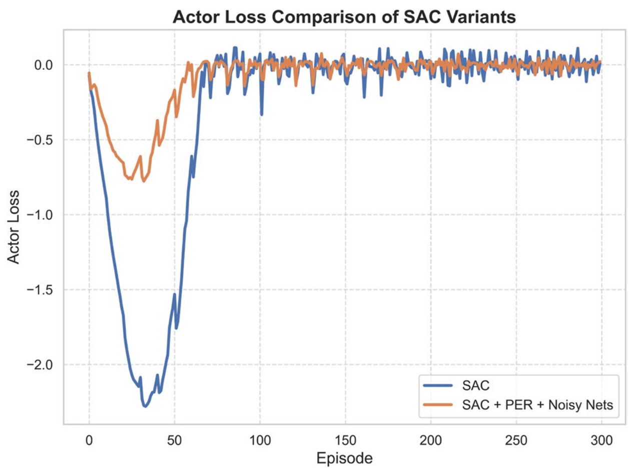

Fig. 7. Actor loss convergence of SAC variants.

scheduling strategies. These findings underscore the superiority and application potential of the proposed approach in solving the economic dispatch problem.

Comparative analysis of RL controllers with enhancement techniques

This study evaluates the generality of the proposed improvement techniques by conducting experiments on three distinct RL controllers: DDPG, SAC, and MPO. Each controller was evaluated in its baseline configuration as well as after incorporating Noisy Nets and PER. Table 4 summarizes the performance metrics, including

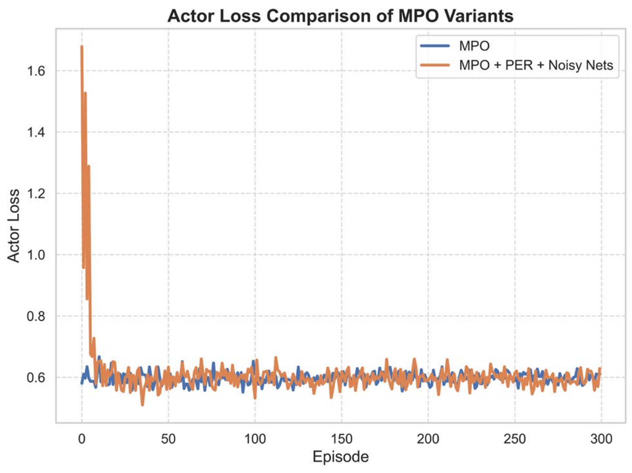

Fig. 8. Actor loss convergence of MPO variants.

training and testing operation costs and system unbalance. The convergence of the actor loss for DDPG, SAC, and MPO is illustrated in Figs. 6, 7, and 8, respectively.

A detailed analysis of the results for each RL controller is presented in the following subsections.

DDPG with enhancement techniques
DDPG serves as the foundational model for TD3 and is inherently deterministic. As shown in Fig. 6, similar to TD3, the model converged successfully after experiencing some initial fluctuations during the early stages of training. The integration of Noisy Nets and PER into DDPG led to a reduction in training operation cost by 11.20% and testing operation cost by 3.72% compared to the baseline DDPG. Notably, the testing unbalance metric showed a substantial improvement, decreasing by 59.44%. It is also worth emphasizing that both the original and enhanced DDPG models achieved a training unbalance of 0 kW. This outcome can be attributed to the deterministic nature of the policy, which enables precise action selection and facilitates effective management of system unbalance. In general, deterministic models are well-suited for handling such constraints, provided that the model has sufficient complexity. However, when model capacity is inadequate, even deterministic policies may face difficulties. In contrast, stochastic policies, which generate action distributions rather than fixed outputs, often struggle to achieve highly precise control.

Although the improvement in operation cost during testing was less pronounced than in the training phase, the enhanced model benefited from a more comprehensive state exploration during training. This led to a marked improvement in the testing unbalance performance, underscoring the importance of robust exploration in achieving long-term stability.

SAC with enhancement techniques
As a stochastic policy algorithm, SAC demonstrates competitive operational cost performance comparable to TD3. As shown in Fig. 7, the model converged after passing through an initial dip in actor loss. The negative values of the actor loss are a result of its definition in SAC and are considered normal. In its baseline configuration, SAC achieved a lower operation cost than TD3. However, because its policy outputs a probability distribution over actions, its ability to perform fine-grained control in specific states is limited. This limitation is reflected in the higher testing unbalance observed with SAC.

The integration of Noisy Nets and PER with SAC resulted in a reduction of the training operation cost by 23.28% and the testing operation cost by 12.45%. Despite the stochastic policy, the incorporation of these techniques yielded significant improvements in economic efficiency. It is worth noting that the training unbalance for the improved SAC model increased relative to the baseline. A possible explanation is that, during certain episodes, the actor's stochastic policy might have incidentally produced actions that coincided with favorable grid conditions, thereby minimizing unbalance. However, such favorable instances are not consistently reproducible across different episodes or unseen test datasets. Nevertheless, the overall effect of incorporating Noisy Nets and PER was positive, as evidenced by the reduction in the cumulative unbalance during the testing period. These findings support the conclusion that the improvement techniques enhance the performance of stochastic models in terms of both cost reduction and system balance.

Although SAC achieves favorable operational cost performance, its relatively high grid unbalance during operation limits its suitability as a reliable solver for regional power grid EDP.

MPO with enhancement techniques
As shown in Fig. 8, the actor loss of MPO converged rapidly during training on this EDP environment. MPO is known for its training stability, especially in high-dimensional action spaces, and consistently converged to an operation cost of approximately 4000 dollars during training. However, in the context of power system dispatch, its overall performance in terms of cost efficiency and system unbalance was less competitive compared to other models. This can be attributed to the design focus of MPO, which emphasizes stable policy updates in complex

Model	Baseline time (min)	With PER + Noisy Nets (min)	Time increase (%)	Number of episodes
TD3	61.00	85.00	39.34	450
DDPG	26.25	68.00	159.05	300
SAC	50.67	67.00	32.16	300
MPO	29.33	55.50	89.22	300

Table 5. Comparison of Training time for baseline models and models with PER + Noisy Nets.

environments rather than direct optimization of global performance. Furthermore, as a stochastic policy method, MPO produces probabilistic action outputs, which can limit its ability to make fine-grained control adjustments in scenarios where high precision is required.

In the generation dispatch problem for electric power systems, the integration of Noisy Nets and PER into RL controllers consistently improved performance. Both deterministic and stochastic policy methods experienced reductions in overall operating costs and enhancements in system balance during testing. Although each algorithm exhibited different levels of improvement across various performance metrics, the collective evidence strongly indicates that these improvement techniques contribute positively to the robustness and generalization capability of RL models when applied to economic dispatch problems.

Among all the methods examined in this study, the deterministic approach, specifically TD3 enhanced with Noisy Nets and PER, demonstrated the best performance, achieving the lowest operation costs and power unbalance values in both training and testing environments. This outcome highlights the effectiveness of combining deterministic policy frameworks with advanced exploration and sampling strategies, particularly in applications requiring high precision and reliability.

Computational time analysis

Table 5 compares the training time of baseline models with their counterparts enhanced by PER and Noisy Nets. The results indicate that while these enhancements consistently improve model performance, they also introduce a noticeable increase in training time. The magnitude of this increase varies across algorithms: DDPG exhibits the largest increase of approximately 159%, whereas SAC records the smallest at around 32%.

The variation in time overhead can be attributed to differences in algorithmic architecture, sensitivity to stochastic perturbations, and network update frequency. DDPG, with its relatively simple structure and high susceptibility to noisy inputs, incurs substantial computational costs when both PER and Noisy Nets are applied. Conversely, TD3 and SAC incorporate stabilization mechanisms, including target policy smoothing and delayed policy updates. These mechanisms mitigate the impact of the enhancements, leading to more moderate increases in training time. The additional computational burden of PER, including priority value computation, sampling distribution maintenance, and replay buffer updates, affects algorithms to different extents depending on the efficiency of their implementation and the structure of their replay mechanisms. These factors collectively explain the observed variation in training time across models.

It should be noted that the reported values reflect single training runs and do not include the extra time required for hyperparameter tuning. The introduction of PER and Noisy Nets expands the hyperparameter search space substantially, potentially prolonging the tuning process, especially under constrained computational resources.

Despite these increases, the proposed method remains practical for power system applications. The additional training time is within acceptable limits and is justified by the performance gains achieved, making the approach suitable for scenarios where a balance between solution quality and computational cost is critical. Furthermore, the inference latency (i.e., the time required for the model to produce a decision, also known as response time) exhibits no significant variation across different models and environments[39] and consistently remains at the millisecond level, which is sufficiently fast. Therefore, the extended training phase does not hinder the applicability of the method in real-time operational contexts.

Discussion and future research

The numerical results presented above indicate that incorporating RL improvement techniques can improve the performance of RL controllers for economic dispatch in power systems. Although the environment utilized in this study is relatively simple (where even the plain TD3 model yields satisfactory solutions), the application of PER and Noisy Nets, along with careful hyperparameter tuning, brings performance close to global optimality[22]. In this setting, more complex controllers appear to offer limited additional benefits. They also tend to complicate hyperparameter tuning and increase training time.

A key challenge in this study is the expansion of hyperparameters introduced by the enhancement techniques. The resulting parameter space is large, making exhaustive grid search impractical. In practice, both academic and industrial work often fixes most settings to well-established defaults[28] and tunes only a small subset that is most sensitive to performance. Although a systematic tuning framework was not the focus, the chosen settings follow established RL practice, supporting fairness and reproducibility.

This consideration explains the decision not to include more advanced algorithms such as Rainbow[28] and DreamerV3[40] in the present experiments. These methods are substantially more complex and are typically designed for higher-dimensional or perception-heavy tasks, which differ from the EDP formulation studied here. Their training is also computationally demanding. In practice, effective use often relies on adaptive hyperparameter tuning or meta-learning procedures; without such support, manual tuning can be slow and unreliable. Evaluation of these high-complexity models is deferred to future work when additional computational resources become available.

Commercial solvers such as CPLEX[41] have been widely applied to EDPs through branch-and-bound or mixed-integer programming formulations. They are capable of producing near-exact solutions in relatively small-scale or simplified settings. However, their computational burden grows rapidly with the number of generators, constraints, and time intervals. This makes them less practical for large-scale or real-time applications. In contrast, the reinforcement learning framework employed in this study avoids repeatedly solving large optimization problems. Once trained, the controller generates decisions within milliseconds. While solver-based approaches may deliver slightly higher accuracy in offline conditions, the RL-based method provides superior adaptability, faster inference speed, and robustness under dynamic operating conditions. These properties are essential for online power system operation.

Another challenge lies in the fundamental differences between RL environments used in gaming and those in power system control. In gaming applications, the state space is typically represented by high-dimensional image frames. In power system scenarios, the state consists of key system parameters computed in real time. As the scale of power system models increases, the computational cost associated with agent–environment interactions grows substantially. This is also the reason why additional power system–related parameters, including CO_2 emissions and frequency deviation, were not included in the current experiments. Incorporating these parameters would significantly increase the computational time required for model–environment interactions. A practical mitigation is to parallelize the RL environment on multi-core CPU architectures. This approach reduces the computational burden.

At present, the proposed TD3 model enhanced with PER and Noisy Nets demonstrates satisfactory performance on the EDP problem within existing regional power grids. In more complex scenarios, such as regional grids characterized by dynamic pricing and stochastic topologies, retraining the controller for the new environment is expected to produce effective results. This highlights the adaptability and generalization potential of the proposed approach across diverse power-system configurations. When the controller is neural network-based, its capacity should be matched to task complexity. Insufficient capacity risks underfitting, whereas excessive capacity and weak regularization can lead to overfitting.

Beyond the EDP, future research might extend RL-based controllers to tackle even more challenging problems, such as stability control in complex power systems, blackout prevention, and management under extreme conditions. In such scenarios, advanced RL frameworks (e.g., those based on the Rainbow architecture) may be required to accommodate the diverse constraints and operational conditions present in large-scale power systems.

Conclusions

This study proposes a novel approach, TD3+PER+Noisy Nets, which integrates RL improvement techniques to address the EDP in power systems. Unlike methods from past research that combine RL models with external approaches, such as MIP[22], to enhance performance, the proposed pipeline directly refines the RL model's workflow through internal architectural modifications. By incorporating Noisy Nets, the agent's exploration capability is significantly enhanced, enabling more effective search within the action space. Simultaneously, the utilization of PER improves sample efficiency and accelerates the training process. Experimental results demonstrate that the proposed method substantially reduces operational costs while enhancing power balance during both training and testing phases.

When compared with the baseline plain TD3 model, the TD3+PER+Noisy Nets model achieves a 54.6% reduction in testing operation cost and a 95.3% reduction in cumulative testing unbalance. Comparative analyses across various RL models, including both deterministic and stochastic policy methods, further confirm the general effectiveness of the improvement techniques. Among all evaluated methods, the deterministic TD3+PER+Noisy Nets model exhibited the best overall performance, achieving the lowest operation cost and power unbalance in both training and testing environments.

In summary, the proposed method provides an alternative to hybrid approaches by leveraging targeted improvement techniques to enhance controller performance for specific power system problems. This study not only establishes a solid theoretical foundation but also offers practical evidence for the successful integration of RL improvement techniques in solving complex optimization problems in power systems. Future research should focus on adaptive hyperparameter tuning, parallelized environment simulations, and extending these techniques to more challenging control tasks, such as stability management and blackout prevention, thereby paving the way for more robust and efficient RL-based controllers in the energy sector.

Data availability

The datasets and environment analysed during the current study are available in the GitHub repository: https://github.com/EnergyQuantResearch/Optimal-Energy-System-Scheduling-Combining-Mixed-Integer-Programming-and-Deep-Reinforcement-Learning.

Received: 26 August 2025; Accepted: 3 November 2025
Published online: 10 December 2025

References

1. Kunya, A. B., Abubakar, A. S. & Yusuf, S. S. Review of economic dispatch in multi-area power system: State-of-the-art and future prospective. *Electric. Power Syst. Res.* **217**, 109089. https://doi.org/10.1016/j.epsr.2022.109089 (2023).
2. Marzbani, F. & Abdelfatah, A. Economic dispatch optimization strategies and problem formulation: A comprehensive review. *Energies* **17**, 550. https://doi.org/10.3390/en17030550 (2024).

3. Fan, J.-Y. & Zhang, L. Real-time economic dispatch with line flow and emission constraints using quadratic programming. *IEEE Trans. Power Syst.* **13**, 320–325. https://doi.org/10.1109/59.667345 (1998).
4. Parikh, J. & Chattopadhyay, D. A multi-area linear programming approach for analysis of economic operation of the indian power system. *IEEE Trans. Power Syst.* **11**, 52–58. https://doi.org/10.1109/59.485985 (1996).
5. Affijulla, S. & Chauhan, S. A new intelligence solution for power system economic load dispatch. In *2011 10th International Conference on Environment and Electrical Engineering*, 1–5, (2011). https://doi.org/10.1109/EEEIC.2011.5874614
6. Aman, M., Jasmon, G., Bakar, A. & Mokhlis, H. A new approach for optimum simultaneous multi-dg distributed generation units placement and sizing based on maximization of system loadability using HPSO (hybrid particle swarm optimization) algorithm. *Energy* **66**, 202–215. https://doi.org/10.1016/j.energy.2013.12.037 (2014).
7. Kahourzade, S., Mahmoudi, A. & Mokhlis, H. B. A comparative study of multi-objective optimal power flow based on particle swarm, evolutionary programming, and genetic algorithm. *Electric. Eng.* **97**, 1–12. https://doi.org/10.1007/s00202-014-0307-0 (2015).
8. Cho, J.-H., Wang, Y., Chen, I.-R., Chan, K. S. & Swami, A. A survey on modeling and optimizing multi-objective systems. *IEEE Commun. Surv. Tutor.* **19**, 1867–1901. https://doi.org/10.1109/COMST.2017.2698366 (2017).
9. Gendreau, M. & Potvin, J.-Y. *Handbook of Metaheuristics* 2nd edn. (Springer, New York, 2010).
10. Vishwakarma, K. K., Dubey, H. M., Pandit, M. & Panigrahi, B. K. Simulated annealing approach for solving economic load dispatch problems with valve point loading effects. *Int. J. Eng. Sci. Technol.* **4**, 60–72. https://doi.org/10.4314/ijest.v4i4.7 (2012).
11. Awadallah, M. A. et al. Memetic salp swarm algorithm for economic load dispatch problems. *Sci. Rep.* **15**, 30539 (2025).
12. Shaban, A. E., Ismaeel, A. A. K., Farhan, A., Said, M. & El-Rifaie, A. M. Growth optimizer algorithm for economic load dispatch problem: Analysis and evaluation. *Processes* **12**, 2593. https://doi.org/10.3390/pr12112593 (2024).
13. Fahim, K. E., De Silva, L. C., Andiappan, V., Shezan, S. A. & Yassin, H. A novel hybrid algorithm for solving economic load dispatch in power systems. *Int. J. Energy Res.* **2024**, 8420107 (2024).
14. Hua, H., Qin, Y., Hao, C. & Cao, J. Optimal energy management strategies for energy internet via deep reinforcement learning approach. *Appl. Energy* **239**, 598–609. https://doi.org/10.1016/j.apenergy.2019.01.145 (2019).
15. Radac, M.-B. & Chirla, D.-P. Near real-time online reinforcement learning with synchronous or asynchronous updates. *Sci. Rep.* **15**, 17158. https://doi.org/10.1038/s41598-025-00492-7 (2025).
16. Cai, M., Xiao, S., Li, J. & Kan, Z. Safe reinforcement learning under temporal logic with reward design and quantum action selection. *Sci. Rep.* **13**, 1925. https://doi.org/10.1038/s41598-023-28582-4 (2023).
17. Tightiz, L. & Yoo, J. A robust energy management system for korean green islands project. *Sci. Rep.* **12**, 22005. https://doi.org/10.1038/s41598-022-25096-3 (2022).
18. Liu, G., Liu, J. & Liu, A. Mitigating sub-synchronous oscillation using intelligent damping control of dfig based on improved td3 algorithm with knowledge fusion. *Sci. Rep.* **14**, 14692. https://doi.org/10.1038/s41598-024-65372-y (2024).
19. Zhang, Z., Zhang, D. & Qiu, R. C. Deep reinforcement learning for power system applications: An overview. *CSEE J. Power Energy Syst.* **6**, 213–225. https://doi.org/10.17775/CSEEJPES.2019.00920 (2020).
20. Chiş, A., Lundén, J. & Koivunen, V. Reinforcement learning-based plug-in electric vehicle charging with forecasted price. *IEEE Trans. Vehicular Technol.* **66**, 3674–3684. https://doi.org/10.1109/TVT.2016.2603536 (2017).
21. Riedmiller, M. Neural fitted q iteration - first experiences with a data efficient neural reinforcement learning method. In *Lecture Notes in Computer Science: Machine Learning: ECML 2005* Vol. 3720 (eds Gama, J. et al.) 317–328 (Springer, Berlin, Heidelberg, 2005).
22. Shengren, H., Vergara, P. P., Salazar Duque, E. M. & Palensky, P. Optimal energy system scheduling using a constraint-aware reinforcement learning algorithm. *Int. J. Electric. Power Energy Syst.* **152**, 109230. https://doi.org/10.1016/j.ijepes.2023.109230 (2023).
23. Claessens, B. J., Vrancx, P. & Ruelens, F. Convolutional neural networks for automatic state-time feature extraction in reinforcement learning applied to residential load control. *IEEE Trans. Smart Grid* **9**, 3259–3269. https://doi.org/10.1109/TSG.2016.2629450 (2018).
24. Wu, H., Xu, Z. & Wang, M. Unrolled spatiotemporal graph convolutional network for distribution system state estimation and forecasting. *IEEE Trans. Sustain. Energy* **14**, 297–308. https://doi.org/10.1109/TSTE.2022.3211706 (2023).
25. Wu, H., Xu, Z., Wang, M. & Jia, Y. Full-model-free adaptive graph deep deterministic policy gradient model for multi-terminal soft open point voltage control in distribution systems. *J. Modern Power Syst. Clean Energy* **12**, 1893–1904. https://doi.org/10.35833/MPCE.2024.000177 (2024).
26. Wu, H. & Xu, Z. Prototype federated reinforcement learning for voltage regulation in distribution systems with physics-aware spatial-temporal graph perception. *IEEE Trans. Sustain. Energy* https://doi.org/10.1109/TSTE.2025.3581286 (2025).
27. Retzlaff, C. O. et al. Human-in-the-loop reinforcement learning: A survey and position on requirements, challenges, and opportunities. *J. Artif. Intell. Res.* **79**, 359–415. https://doi.org/10.1613/jair.1.15348 (2024).
28. Hessel, M. *et al.* Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, https://doi.org/10.1609/aaai.v32i1.11796(2018).
29. Vinyals, O. et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature* **575**, 350–354. https://doi.org/10.1038/s41586-019-1724-z (2019).
30. Mnih, V. *et al.* Playing atari with deep reinforcement learning (2013). ArXiv preprint arXiv:1312.5602
31. Mnih, V. et al. Human-level control through deep reinforcement learning. *Nature* **518**, 529–533. https://doi.org/10.1038/nature14236 (2015).
32. Grondman, I., Busoniu, L., Lopes, G. A. D. & Babuska, R. A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *IEEE Trans. Syst., Man, Cyber., Part C (Appl. Rev.)* **42**, 1291–1307. https://doi.org/10.1109/TSMCC.2012.2218595 (2012).
33. Lillicrap, T. P. *et al.* Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971 (2015).
34. Haarnoja, T., Zhou, A., Abbeel, P. & Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning*, 1861–1870, (PMLR, 2018). https://doi.org/10.48550/arXiv.1801.01290
35. Abdolmaleki, A. *et al.* Maximum a posteriori policy optimisation (2018). ArXiv preprint arXiv:1806.06920
36. Fujimoto, S., Hoof, H. & Meger, D. Addressing function approximation error in actor-critic methods. In *Proceedings of the 35th International Conference on Machine Learning*, 1587–1596, (PMLR, 2018). https://doi.org/10.48550/arXiv.1802.09477
37. Schaul, T., Quan, J., Antonoglou, I. & Silver, D. Prioritized experience replay (2016). arXiv:1511.05952.
38. Fortunato, M. *et al.* Noisy networks for exploration (2019). arXiv:1706.10295.
39. Smirnov, I. & Gu, S. Rlbenchnet: The right network for the right reinforcement learning task (2025). arXiv:2505.15040.
40. Hafner, D., Pasukonis, J., Ba, J. & Lillicrap, T. Mastering diverse domains through world models (2024). arXiv:2301.04104.
41. Cplex, I. I. V12. 1: User's manual for cplex. *Int. Bus. Mach. Corporat.* **46**, 157 (2009).

Author contributions

C.X. conceived the experiments, conducted the experiments, analysed the results, and wrote the manuscript. N.H. and M.I. contributed to the experimental design and reviewed the manuscript. W.J.K.R., H.B.M., and H.A.B.I. reviewed and revised the manuscript. All authors approved the final version of the manuscript.

Funding
This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Declarations

Competing interests
The authors declare no competing interests.

Additional information
Correspondence and requests for materials should be addressed to C.X.