

ORIGINAL RESEARCH

Open access

Dynamic Security Margin Management via Deep Reinforcement Learning

Samuel Boateng^{1*}, Kwesi Mensah¹, Kojo Asante², Linda Owusu¹

Abstract

This paper presents a real-time preventive control approach utilizing deep reinforcement learning (DRL) to identify optimal control actions that maintain an adequate secure operating limit (SOL) in power systems. The SOL represents the boundary of the most stressed pre-contingency operating condition that the system can endure following a specified set of credible contingencies while satisfying predefined stability requirements. The proposed DRL technique employs a hybrid control framework that enables simultaneous management of both discrete and continuous control variables. Its effectiveness is assessed using a modified Nordic32 test system. Findings indicate that the DRL method rapidly acquires a robust control policy capable of securing sufficient SOL across various operating conditions. The approach is benchmarked against a rule-based lookup table method and a DRL variant restricted to discrete actions. Results demonstrate that the hybrid DRL controller consistently outperforms the alternatives across all evaluated test cases, highlighting the advantages of jointly optimizing discrete and continuous variables for greater flexibility and operational efficiency.

Keywords Deep reinforcement learning, Preventive control, Proximal policy optimization, Secure operating limit

*Correspondence:

Samuel Boateng

samuel.boateng@ucc.edu.gh

¹ Department of Electrical Power Systems, Faculty of Engineering, University of Ghana, Accra, Ghana

² Department of Energy Data Science, Faculty of Engineering, Kwame Nkrumah University of Science and Technology, Kumasi, Ghana

Introduction

An electric power system is considered to operate securely if it remains stable after the outage of any single element, as defined by the N-1 criterion. To uphold security, operators routinely calculate security margins and implement preventive measures whenever these margins fall below acceptable thresholds. This study focuses on determining the most suitable control actions to maintain an adequate secure operating limit (SOL). Defined as the distance to the most heavily loaded pre-contingency state that can tolerate a given set of plausible contingencies without breaching stability limits [1], the SOL offers a more accurate reflection of dynamic security than traditional static evaluation techniques. Its calculation, however, is computationally intensive, typically demanding time-domain simulations (or quasi-steady-state approximations) to capture the system's dynamic behavior following disturbances.

Common preventive controls for enhancing security margins involve (1) reactive power management and voltage regulation via transformer tap adjustments and capacitor or reactor switching, and (2) redispatching generation or shedding load to alleviate overloads on critical transmission lines and buses [2]. Historically, such stability controls have relied on predefined lookup tables derived from extensive offline studies covering standard operating scenarios. Nevertheless, because different control actions carry varying economic costs, these heuristic approaches often lead to suboptimal decisions that reduce overall system efficiency.

Several investigations have explored optimal control strategies for power system stability. Yet, power systems are characterized by high dimensionality, strong nonlinearities, and inherent uncertainties [3]. Achieving real-time optimal solutions usually necessitates model simplifications or linearizations [2]. Alternative data-driven techniques, such as those in Cai et al. [4] and Li et al. [5], involve offline evaluation of numerous state-action combinations, with the resulting optimal actions stored in a database and approximated via supervised learning models. However, exhaustively exploring all possible state-action pairs becomes impractical when the action space is large, leading to prolonged training times and limited adaptability to system changes, such as topology modifications.

Reinforcement learning (RL) has recently emerged as a powerful paradigm for tackling intricate control challenges by allowing an agent to learn optimal behavior through trial-and-error interactions with the environment or its simulator [6]. When combined with deep neural networks, deep reinforcement learning (DRL) has demonstrated remarkable success across domains such as gaming [7, 8], self-driving vehicles [9], and robotic manipulation [10]. DRL's ability to automatically extract relevant features from high-dimensional inputs makes it particularly suitable for power system applications involving numerous states and control variables. Prior applications of RL to power system stability control have predominantly addressed normal or emergency operating states [11], covering areas such as automatic voltage regulation using traditional RL [12] or DRL [13], frequency control [14], emergency load shedding [15], transient stability enhancement [16], and power oscillation damping [17].

Literature on RL-based preventive control—specifically aimed at sustaining secure N-1 operation—is comparatively scarce. According to Glavic [11], this may stem from the traditional formulation of such problems as static optimization tasks typically addressed by classical methods. Notable exceptions include Zarrabian et al. [18], which developed an RL approach for preventive active power rescheduling to mitigate cascading outages and blackouts. In Zhou et al. [19], a DRL framework was introduced for optimal generation rescheduling in distribution networks to improve resilience. Another study by Dong et al. [20] proposed a multi-objective RL method to adjust generator setpoints and compensation devices to enhance short-term voltage stability. Additionally, Li et al. [21] presented a DRL technique for selecting security control strategies, with emphasis on angle stability. To our knowledge, no prior DRL implementation has specifically targeted the restoration of sufficient security margins through preventive control actions.

Effective preventive control often requires the concurrent adjustment of discrete actions (such as switching shunt capacitors) and continuous actions (such as the magnitude of generation redispatch or load shedding). However, most established DRL algorithms—including deep Q-networks (DQN) and deep deterministic policy gradient (DDPG)—are tailored for either purely discrete or purely continuous action spaces. A common workaround involves converting all variables into a single type, for example, by discretizing continuous controls or treating discrete actions as continuous with arbitrary thresholds [22]. While functional in some cases, these conversions can degrade performance and complicate the learning process. Recent works have introduced extensions to support hybrid action spaces. For instance, Li et al. [23] developed a real-time demand response strategy that jointly optimizes discrete and continuous decisions. At the same time, Qiu et al. [24] proposed a multi-agent RL method for simultaneously controlling discrete and continuous ancillary services from electric vehicles.

The present work addresses a gap in preventive control methodologies by proposing a novel DRL-based framework that delivers real-time optimal actions to maintain an adequate SOL. It further investigates the constraints imposed by different action space representations and introduces a hybrid DRL controller that manages both discrete and continuous variables in tandem. The primary contributions are as follows:

- A DRL-based preventive control solution that enables operators to continuously track and regulate the dynamic security margin represented by the SOL, a capability unattainable in real time with conventional optimization tools. The problem is reformulated as a dynamic learning framework suitable for DRL. The agent observes system states from measurements; when the security margin becomes insufficient, it recommends optimized control actions to restore secure operation. Should initial actions prove inadequate, the process iterates by re-evaluating the updated state and applying further corrections until the desired security level is achieved.

- A DRL architecture grounded in the proximal policy optimization (PPO) algorithm that natively supports hybrid action spaces combining continuous and discrete controls. This hybrid scheme is systematically compared against conventional single-action-space approaches, including rule-based methods. The ability to handle mixed action types allows the policy to leverage a broader range of control devices, resulting in more efficient restoration of security margins.
- Comprehensive performance analysis covering robustness against varying operating scenarios and resilience to measurement noise.

Reinforcement Learning

Reinforcement learning (RL) provides a data-driven framework for addressing complex sequential decision-making and optimal control problems. RL tasks are typically formulated as discrete-time Markov decision processes (MDPs). In this setting, an agent applies a policy to interact with the environment, generating sequences of states, actions, and associated rewards. The reward signal provides feedback on the quality of the chosen actions. The primary goal of the agent is to learn a policy that maximizes the expected cumulative discounted reward over time. This is achieved through repeated interaction with the environment or its simulator [25].

In the present work, the MDP is defined by the following components: a state space S , a hybrid action space A containing both discrete and continuous actions, an initial state distribution $p_1(s_1)$, a stationary transition probability distribution $p(s_{t+1}|s_t, a_t)$ that obeys the Markov property $p(s_{t+1}|s_1, a_1, \dots, s_t, a_t) = p(s_{t+1}|s_t, a_t)$, and a reward function $R: S \times A \rightarrow \mathbb{R}$. The symbols $a(t)$, $s(t)$, and $r(t)$ denote the action, state, and reward at time step t , respectively [25].

A parameterized policy is employed to select actions within the MDP. Here, the policy is stochastic and expressed as $\pi_\theta(a_t | s_t): S \rightarrow P(A)$, where $P(A)$ represents the set of probability distributions over the action space A , $\theta \in \mathbb{R}^n$ is the vector of policy parameters, and $\pi_\theta(a_t | s_t)$ gives the conditional probability of selecting action a_t in state s_t .

The return G_t is the discounted sum of rewards starting from time t onward, defined as:

$$G_t^{\gamma} = \sum_{k=t}^{\infty} \gamma^{k-t} R(s_k, a_k) \quad (1)$$

Where $0 < \gamma < 1$ is the discount factor.

The state-value function $V^{\pi}(s)$ represents the expected return when starting in state s and following policy π thereafter: $V^{\pi}(s) = E[G_t^{\gamma} | s_t = s; \pi]$.

The action-value function $Q^{\pi}(s, a)$ is the expected return when taking action a in state s and then continuing according to policy π : $Q^{\pi}(s, a) = E[G_t^{\gamma} | s_t = s, a_t = a; \pi]$.

Policy gradients and actor–critic methods

Policy gradient methods are a family of model-free RL algorithms that directly optimize a parameterized policy without requiring an intermediate value function [26]. These techniques aim to maximize an objective function $J(\theta)$ with respect to the policy parameters θ . Updates are typically performed via stochastic gradient ascent:

$$\theta_{t+1} = \theta_t + \alpha \nabla J(\theta_t) \quad (2)$$

Where $\nabla J(\theta_t)$ is a stochastic estimate of the policy gradient, and α is the learning rate.

A widely used gradient estimator takes the form:

$$\widehat{\nabla J(\theta_t)} = \widehat{\mathbb{E}}_t \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \widehat{A}_t \quad (3)$$

Where $\widehat{\mathbb{E}}_t[\dots]$ denotes an empirical average computed over a batch of sampled transitions, $\nabla_{\theta} \log(\pi_{\theta}(a_t | s_t))$ is the gradient of the log-probability of the selected action under the current policy, and $\widehat{A}_t$ is an estimate of the advantage function at time t . The advantage can be expressed as:

$$\widehat{A}_t = \widehat{Q}^{\pi}(s_t, a_t) - \widehat{V}^{\pi}(s_t) \quad (4)$$

In this study, the control tasks are relatively short-horizon episodic problems. This property allows the use of the sampled return G_t (as defined earlier) as an unbiased estimate of the action-value function $\widehat{Q}^{\pi}$, following the REINFORCE algorithm [27].

Since the true value function is unknown, a function approximator parameterized by weights ϕ is employed. The value function is learned concurrently with the policy by minimizing a loss function $L(\phi)$, typically the mean-squared error between the actual value $V^{\pi}(s_t)$ and its approximation $\widehat{V}_{\phi}^{\pi}(s_t)$:

$$L(\phi) = \mathbb{E}_{\pi} V^{\pi}(s_t) - \widehat{V}_{\phi}^{\pi}(s_t)^2 \quad (5)$$

The parameters ϕ are updated by computing the gradient of this loss and applying stochastic gradient descent (or accelerated variants such as RMSprop [28]) on mini-batches of data. Because G_t provides an unbiased estimate of the value, it serves as the target for training the critic. This setup corresponds to an actor–critic architecture, in which the actor learns the policy π_{θ} and the critic estimates the value function $\widehat{V}_{\phi}^{\pi}$. In the current implementation, both the policy parameters θ and the value function parameters ϕ correspond to the weights of two separate neural networks. Training seeks to determine the optimal weights for these networks.

Proximal policy optimization

The Proximal Policy Optimization (PPO) algorithm [29] is adopted in this work. PPO is a policy gradient method known for its straightforward implementation, improved sample efficiency compared with many alternatives, and suitability for hybrid action spaces. This paper employs the clipped variant of PPO, which optimizes the following objective:

$$J^{\text{clip}}(\theta) = \widehat{\mathbb{E}}_t \min r_t(\theta) \widehat{A}_t, \text{clip} r_t(\theta), 1 - \epsilon, 1 + \epsilon \widehat{A}_t \quad (6)$$

Where $r_t(\theta)$ is the probability ratio:

$$r_t = \frac{\pi_{\theta}(a_t | s_t)}{\pi_{\theta_{\text{old}}}(a_t | s_t)} \quad (7)$$

And ϵ defines the clipping threshold. The term θ_{old} refers to the policy parameters used to collect the current batch of transitions before any update.

The clipping mechanism prevents excessively large policy updates, enabling multiple epochs of optimization on the same batch of samples without destabilizing the learning process. At the first epoch, r_t equals 1 because the current policy matches the sampling policy. In subsequent epochs, the policy is encouraged to increase r_t above 1 when the advantage $\hat{A}_t$ is positive (making beneficial actions more likely) and to decrease r_t below 1 when the advantage is negative (reducing the likelihood of poor actions).

Adaptations for continuous-discrete (Hybrid) control

Most advanced RL algorithms are designed to operate with either purely discrete or purely continuous action spaces [22]. A typical workaround involves converting the entire action space into one uniform type—either by discretizing continuous variables or by approximating discrete choices as continuous values using predefined thresholds. Although such transformations can be effective in specific cases, they often reduce performance and increase the difficulty of the learning task.

To accommodate a hybrid action space comprising both continuous and discrete components, this study builds on the approach proposed by Neunert et al. [22] with minor modifications. The hybrid policy $\pi_\theta(a|s)$ is modeled as a state-dependent probability distribution over a joint set of discrete and continuous random variables. Assuming independence across individual action dimensions a_i , the policy factorizes as:

$$\pi_\theta(a|s) = \pi_\theta^c(a^c|s)\pi_\theta^d(a^d|s) = \prod_{a^i \in a^c} \pi_\theta^c(a^i|s) \prod_{a^i \in a^d} \pi_\theta^d(a^i|s) \quad (8)$$

Where a^c and a^d denote the continuous and discrete subsets of the action vector a , respectively.

Each continuous action dimension is modeled as a Gaussian (normal) distribution:

$$\pi_\theta^c(a^i|s) = \mathcal{N}(\mu_{i,\theta}(s), \sigma_{i,\theta}^2(s)) \quad (9)$$

With state-dependent mean $\mu_{i,\theta}(s)$ and variance $\sigma_{i,\theta}^2(s)$.

A Bernoulli distribution represents each discrete action dimension:

$$\pi_\theta^d(a^i|s) = \text{Bernoulli}^i P_\theta^i(s) \quad (10)$$

Where $P_{i,\theta}(s)$ is the state-dependent probability of selecting the discrete action.

In the implementation, the outputs $\mu_{i,\theta}(s)$, $\sigma_{i,\theta}^2(s)$, and $P_{i,\theta}(s)$ are produced by a branched neural network architecture. Further details regarding the hybrid control scheme, the overall DRL architecture,

Proposed Framework for Real-Time Security Margin Control

This section describes the overall framework developed for real-time control of the power system security margin using deep reinforcement learning. The DRL-based tool continuously observes the system state via available measurements and recommends optimized control actions to operators when the secure operating limit (SOL) drops below a predefined threshold. The section also introduces the test system, the precise definition of the SOL, and the detailed implementation of the DRL agent, including neural network design, state representation, action space, reward function, and data generation.

Test system and security margin definitions

All experiments are conducted on a modified version of the Nordic32 test system, as described by Van Cutsem *et al.* [30]. A schematic overview of the system is shown in **Figure 1**.

The secure operating limit (SOL) is defined as the maximum level of stress the system can tolerate in the pre-contingency state while remaining stable after a specified set of credible contingencies, without violating established stability criteria [1]. It quantifies how much additional stress the system can withstand while preserving N–1 security and serves as an indicator of available transmission capacity. Compared with conventional static security assessment methods, the SOL provides a more accurate security margin, particularly in systems with a high proportion of loads exhibiting rapid restoration dynamics [31].

Because the SOL accounts for the system's dynamic behavior following disturbances, its evaluation requires time-domain simulations, quasi-steady-state (QSS) approximations [1], or a combination of both [32].

System stress is generally modeled through increases in load demand and/or adjustments in generation dispatch—variables that operators can monitor and influence in the pre-contingency condition. Security margins are evaluated with respect to a selected set of credible contingencies. To limit computational effort, contingency filtering is applied to retain only those faults that result in low security margins for the current operating point.

For a given contingency, the SOL is computed via an iterative search procedure resembling the binary search method outlined by Van Cutsem *et al.* [1]. The process, illustrated in **Figure 2**, involves progressively narrowing the stress interval by repeatedly assessing system security under a critical fault at varying stress levels. Black markers represent secure states, while white markers indicate insecure states. The diagram shows the procedure starting from a secure operating point. When the initial state is secure, system stress is incrementally increased by ΔP . If the new state remains secure, stress is raised again by ΔP . If it becomes insecure, stress is reduced by $\Delta P/2$. This iterative adjustment continues until a secure operating point is identified and the stress increment becomes sufficiently small.

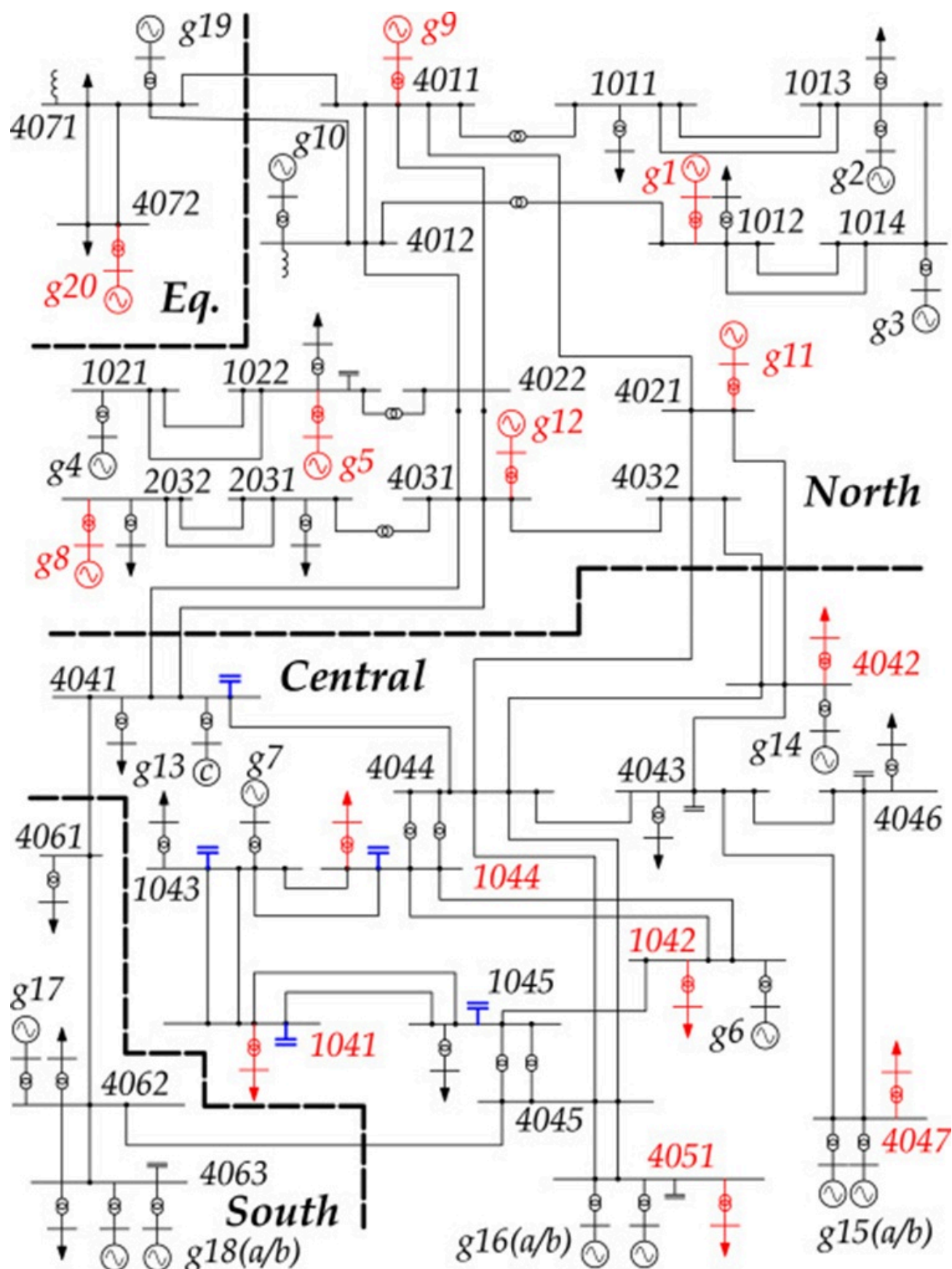


Figure 1. Single-line diagram of the updated Nordic32 test system [30]. Generators and loads that form part of the continuous action space are indicated in red, while shunt capacitors included in the discrete action space are shown in blue. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

In the simulations, the system was progressively stressed by raising the demand in the “Central” region of the modified Nordic32 system, with an equal amount of additional generation supplied from the “North” region. All load power factors were left at their original values. The extra load and generation were allocated proportionally to each bus's base-case loading and each generator's rated power. The dimensioning contingencies examined were the outages of the three largest generators located in the “Central” area: g14, g15, and g16. At every operating condition, the contingency yielding the smallest Security Operating Limit (SOL) was selected as the critical one for that point.

The SOL was calculated using detailed dynamic simulations performed in PSS®E 35.0.0. A simple voltage-based criterion defined system security: the post-contingency system was considered acceptable only if every transmission bus voltage stayed above 0.90 per unit by the end of the simulation period. While the modified Nordic32 system is mainly vulnerable to long-term voltage collapse, other forms of instability (such as transient angular instability leading to generator loss of synchronism and subsequent local voltage dips) could also trigger violations of this threshold. Frequency stability was not directly evaluated and was assumed to be maintained by the generators' automatic frequency regulation. Each dynamic simulation ran for up to 600 seconds but was halted earlier if the system fully collapsed (any bus voltage dropping below 0.7 pu) or reached a stable state before the time limit. This ensured that every case ended with a clear outcome—either stable or unstable.

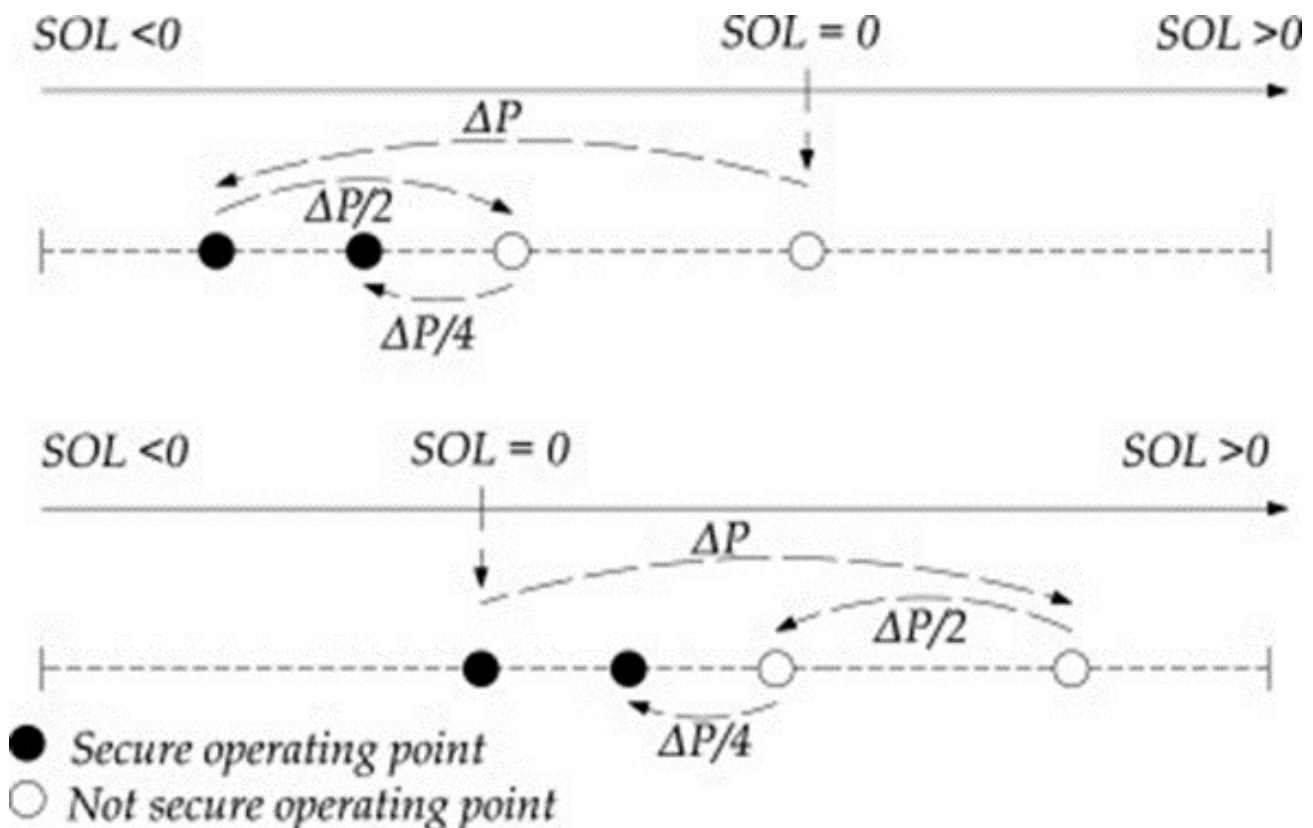


Figure 2. Diagram showing the SOL search procedure for both a secure and an insecure starting operating point.

MDP formulation for security margin control

The problem of controlling the Security Operating Limit (SOL) is modeled as an episodic Markov Decision Process (MDP). At the start of every episode, the deep reinforcement learning (DRL) agent is given a representation of the current system state st based on available measurements. Following the current policy $\pi\theta$, the agent selects actions that are then applied to the power system. These actions, together with the system's transition dynamics, move the system to a new state ($st \rightarrow st_{+1}$) and generate a reward R_t . The episode terminates if the agent successfully raises the SOL above the required threshold or if the maximum allowed number of time steps is reached. Otherwise, the agent continues to receive new observations, select actions, and collect rewards.

The MDP for SOL control is specified as follows:

States: The state representation provided to the actor and critic networks should ideally fulfill the Markov property and contain all necessary information to identify the optimal policy and value function. Since the control task focuses on the pre-contingency steady-state condition, the static network state (bus voltage magnitudes and angles) is theoretically adequate. However, after testing various options, it was determined that a state vector st composed of all bus voltage magnitudes t , together with active and reactive power flows on every transmission line and transformer, produced more reliable training convergence than using voltage magnitudes and angles alone. The current episode time step t was also appended to the state vector. To improve robustness against measurement noise and input variations, each state component was randomly perturbed by multiplying it by a normally distributed value (mean = 1, standard deviation = 0.001). All state variables were then normalized by subtracting the mean and dividing by the standard deviation. These statistics were calculated from a buffer of up to 10,000 previously observed states. Once the buffer reached 10,000 entries, the mean and standard deviation values were frozen for subsequent normalizations.

Actions: The DRL agent can apply both continuous and discrete control actions. Continuous actions adjust generation schedules and implement load shedding to lower the power transfer across the system. This is done by decreasing active power demand at selected load buses in the "Central" area (4042, 4047, 4051, 1041, 1042, 1044) while simultaneously increasing active power output at selected generators in the "North" area (G1, G5, G8, G9, G11, G12, G20). The load reduction at each participating bus is allocated proportionally to its initial loading, with load power factors held constant. Discrete actions consist of switching in an extra 100 MVar of shunt reactive power compensation from any combination of available capacitor banks. The discrete actions (D1–D5) correspond to shunt capacitors located at buses: D1 at 1041, D2 at 1043, D3 at 1044, D4 at 1045, and D5 at 4041. The buses and devices involved in continuous and discrete actions are marked in red and blue, respectively, in the single-line diagram in [Figure 1](#).

State transition: The transition between states is fully deterministic and follows the set of differential-algebraic equations that define the dynamic model implemented in PSS®E.

Rewards: The reward R_t is calculated based on the achieved SOL after the actions, combined with penalties for the continuous actions (C_{cont}) and discrete actions (C_{disc}). Although the reward is dimensionless in this study, in practical deployment, it should correspond to real economic costs and benefits. Activating any discrete action incurs a penalty of -5 to represent mechanical wear on the shunt capacitors. Continuous actions carry a cost of -0.1 per MW of adjusted power transfer (e.g., a ± 200 MW change yields a -20 penalty). This accounts for the economic impact of market redispatch or load/generation curtailment. The primary objective is to bring the SOL to at least 30 MW, ensuring an adequate security margin that satisfies the N-1 criterion with some buffer for uncertainties. If the SOL remains below 30 MW after an action, an additional penalty of -50 is applied. These reward components are designed to encourage the agent to restore sufficient security in the fewest possible steps while using the smallest necessary control effort. The final reward, incorporating the resulting SOL, is then calculated as:

$$R_t = \begin{cases} C_{cont} + C_{disc} + \text{SOL} - 50, & \text{if SOL} \leq 30 \text{ MW} \\ C_{cont} + C_{disc}, & \text{otherwise} \end{cases} \quad (11)$$

Training data generation

An overview of the training data generation process and subsequent DRL agent training is presented in **Figure 3**. The individual steps are described below. To accelerate the process, data generation was parallelized across multiple CPU cores.

1. Generate initial operating conditions: A wide range of initial operating conditions (OCs) was generated for training. Each load in the system was individually scaled by multiplying its active power by a random factor drawn from a uniform distribution between 90% and 130% of the base value, while keeping power factors unchanged. The overall load increase or decrease was balanced by proportionally adjusting generator outputs according to their initial active power contributions. All generated OCs were solved using a full Newton–Raphson power flow; non-convergent cases were discarded and regenerated.
2. Sample state st and actions at : For each initial OC, the current state st was extracted and fed into the actor network. The actor outputs the parameters of the hybrid policy ($a|s$), from which the agent samples the actions to be executed.
3. Take actions and solve load flow: After applying the sampled actions, a new Newton–Raphson load flow was solved to obtain the subsequent system state st_{+1} .
4. Compute the SOL: Once the new operating point was obtained, the SOL was determined using the procedure described previously.
5. End episode and save transitions: The episode concluded if the SOL reached or exceeded 30 MW (the target security margin) or if the time step limit $T = 8$ was reached. At episode termination, the complete sequence of transitions ($st, at, Rt, \dots, sT, aT, RT$) was saved for later training. This data generation cycle was repeated for $N = 64$ episodes before the collected data was used to update the agent.

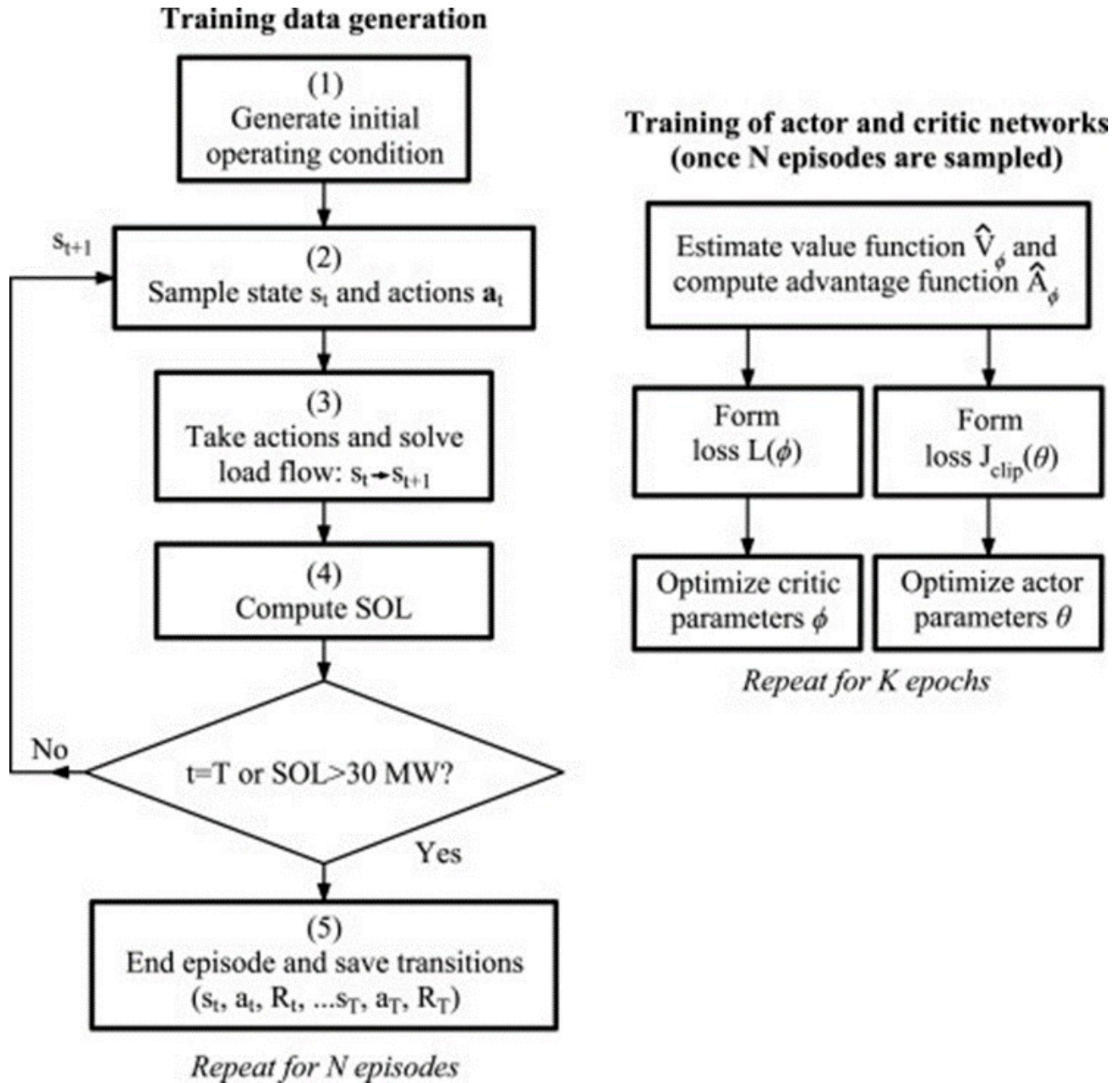


Figure 3. Overview diagram of the training data creation process and the training procedure for the actor and critic networks.

Architecture of the actor and critic networks

The actor network, depicted in [Figure 4](#) and specified in [Table 1](#), begins with a shared hidden layer, followed by dedicated hidden layers for each action type. From these branches, the network generates the necessary parameters to define a Gaussian (Normal) distribution for the continuous actions and Bernoulli distributions for the discrete actions.

Specifically, the continuous actions are sampled from a Normal distribution with mean μ_{cont} and standard deviation σ_{cont} . The mean is produced directly through a linear activation at the output, while the standard deviation is obtained via a Softplus activation to ensure it remains strictly positive. For the discrete actions, each Bernoulli distribution is parameterized by a probability value (bounded between 0 and 1) generated using a sigmoid activation in the final layer.

In contrast, the critic network operates independently from the actor. It is a straightforward feedforward neural network with two hidden layers and a linear output activation, as detailed in Table 1.

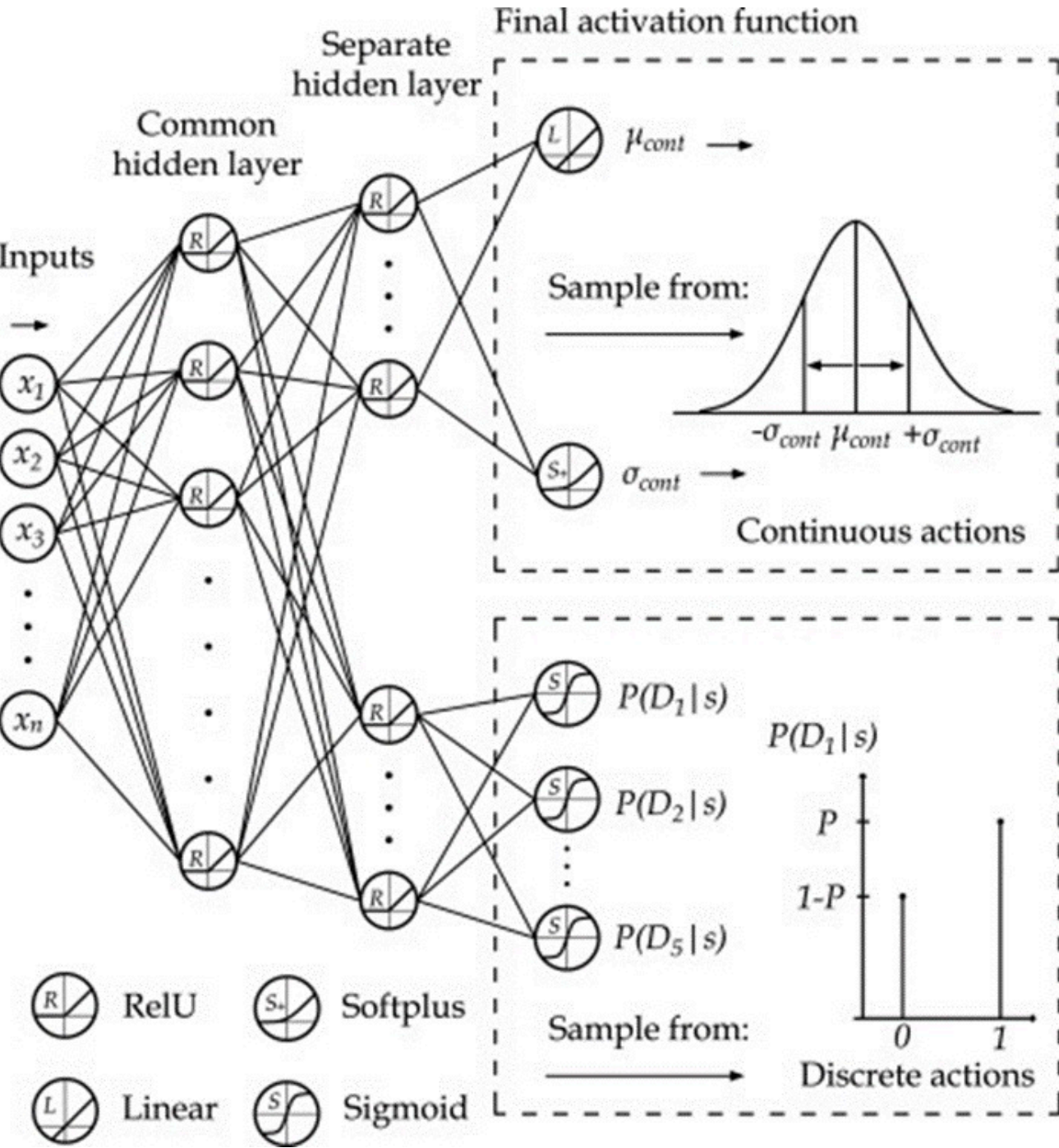


Figure 4. Architecture of the actor network, showing the outputs that form the stochastic continuous-discrete policy.

Table 1. Design and hyperparameters used in training.

Empty cell	Empty cell	Parameter	Values
Architecture	Critic	Number of inputs	499

	Actor	Neurons in the first layer	128
		Neurons in the second hidden layer	64
		Final activation function	Linear
		Hidden layer activation	ReLU
		Number of inputs	499
		Neurons in a common hidden layer	128
		Neurons in each separate hidden layer	64
		Final activation for μ cont	Linear
		Final activation for σ cont	Softplus
		Final activation for $P(Dx s)$	Sigmoid
Empty Cell	Training	Hidden layer activation	ReLU
		Max Epochs (K)	10
		PPO clip parameter (ϵ)	0.2
		Discount factor (γ)	0.99
Empty Cell		Optimizer	RMSprop [28]

Training of the actor and critic networks

After collecting a batch of training samples, both the actor and critic networks were updated. The training process was implemented in Python using TensorFlow, which automatically computed gradients for the loss functions.

First, the critic network estimated the state value function $V_{\phi}\pi$ for every state in the batch. The discounted returns $G_t\gamma$ were calculated according to Eq. 1, using a discount factor $\gamma = 0.99$. Advantage values A_t were then derived using Eq. 4 and the critic's value estimates.

The actor's final objective was obtained by averaging the clipped surrogate objective J_{clip} across all samples from the N episodes. The critic's objective was defined as the mean squared error (MSE) between the predicted values and the target returns, as given by Eq. 5, also averaged over all samples from the N episodes.

These objective functions were optimized using the RMSprop optimizer, which is well-suited for stochastic gradient-based updates. Various optimizers and hyperparameter combinations were tested during development, but only the best-performing configuration is reported in Table 1. Dropout regularization was experimented with, but did not yield any performance gains.

The actor parameters θ were updated to maximize its objective, while the critic parameters ϕ were updated to minimize its objective. Training was performed for $K = 10$ epochs over the entire batch of N episodes. Although an extensive hyperparameter search was performed, further improvements might still be possible by fine-tuning aspects such as learning rates or the number of neurons per layer. To accelerate convergence and improve stability in later training phases, a scheduled learning rate was used, as specified in Table 2.

Table 2. Learning rates over training iterations.

Empty Cell	α_{critic}	α_{actor}
Training iteration ≤ 250	$5 \cdot 10^{-3}$	$1 \cdot 10^{-3}$
Training iteration > 250	$5 \cdot 10^{-4}$	$1 \cdot 10^{-4}$

Results and Discussion

Training outcomes

The hybrid DRL agent underwent 600 training iterations, encompassing 38,400 distinct episodes and a total of 68,900 samples (each episode consisting of up to 8 time steps/samples, depending on the episode's duration). Considerable variation in episode rewards was observed across different episodes. To prevent premature termination of the training process, the mean episode reward was calculated over every 100 episodes. Training was considered converged and subsequently halted if the average reward failed to improve over the preceding 100 episodes.

Figure 5 summarizes the training performance. Sub-figure (i) depicts the total episode reward, sub-figure (ii) illustrates the final SOL achieved in each episode, and sub-figure (iii) shows the number of time steps per episode. A centered moving average, calculated over 500 episodes, is indicated by the red line. For clarity, only every 100th value is displayed during training. The results indicate rapid performance improvement up to approximately 19,000 episodes, after which the policy consistently achieved an SOL above the 30 MW threshold with just a single time step in most episodes. Subsequent enhancements were primarily driven by fine-tuning the level of action activation across different scenarios.

Figure 6 presents the evolution of the policy parameters throughout training. Initially, the standard deviation parameter, σ_{cont} , increased, followed by adjustments to the mean parameter, μ_{cont} . After training on roughly 40,000 samples, μ_{cont} stabilized. Beyond this point, further improvements were largely achieved by reducing exploration, which involved lowering σ_{cont} and decreasing the randomness in discrete actions. Sub-figure (iii) of **Figure 6** shows the probability of executing discrete action D3, indicating that the policy progressively became more confident in deciding whether to activate or skip the discrete action. A similar trend was observed for the other discrete actions.

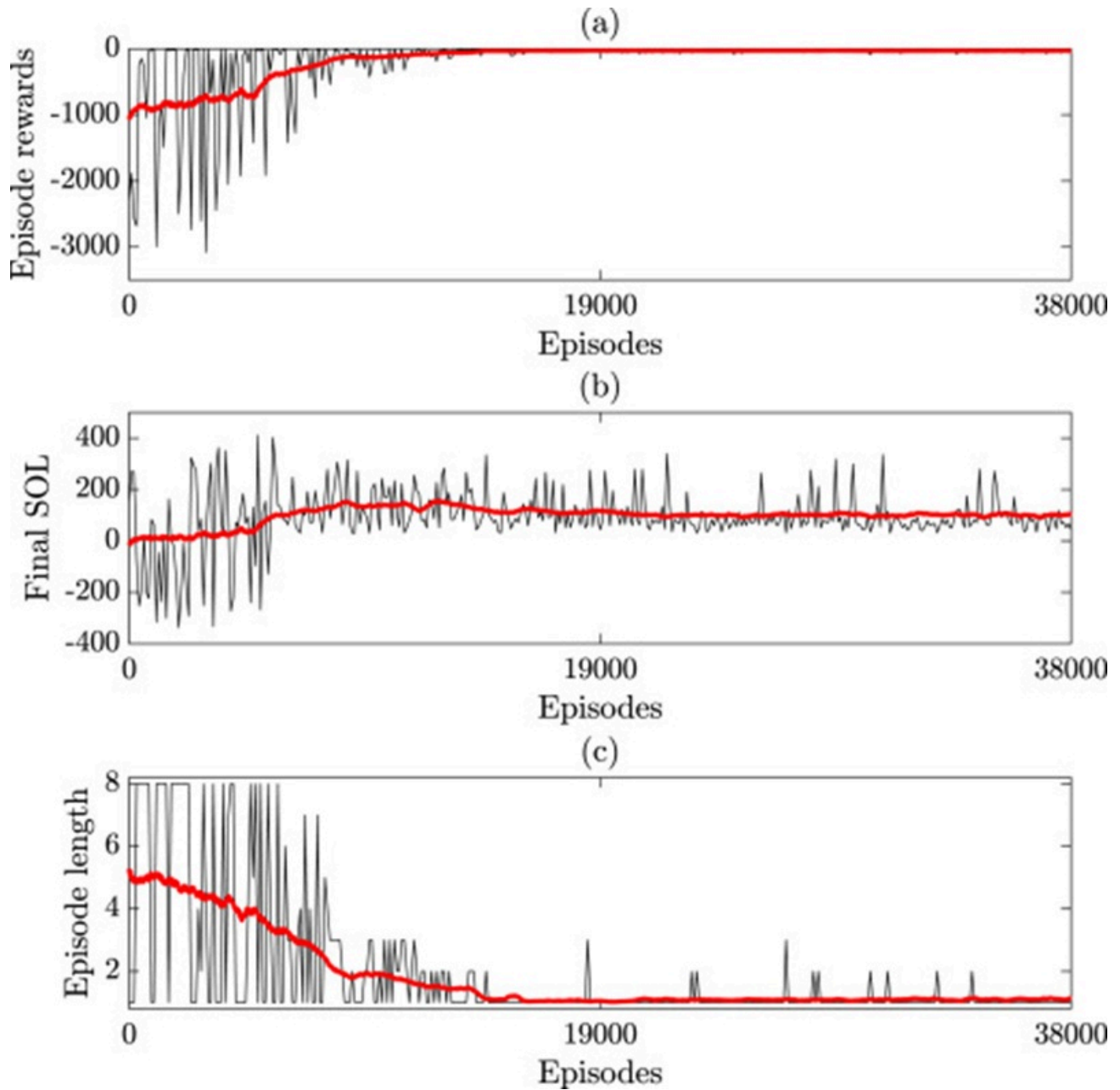


Figure 5. Evolution of performance across training episodes and samples: Sub-figures illustrate (i) the episode rewards throughout all episodes, (ii) the final SOL achieved, and (iii) the length of each episode. The red line represents a moving average calculated over 500 episodes. For clarity, only every 100th value is displayed (For interpretation of the color references in this figure legend, readers are directed to the online version of this article).

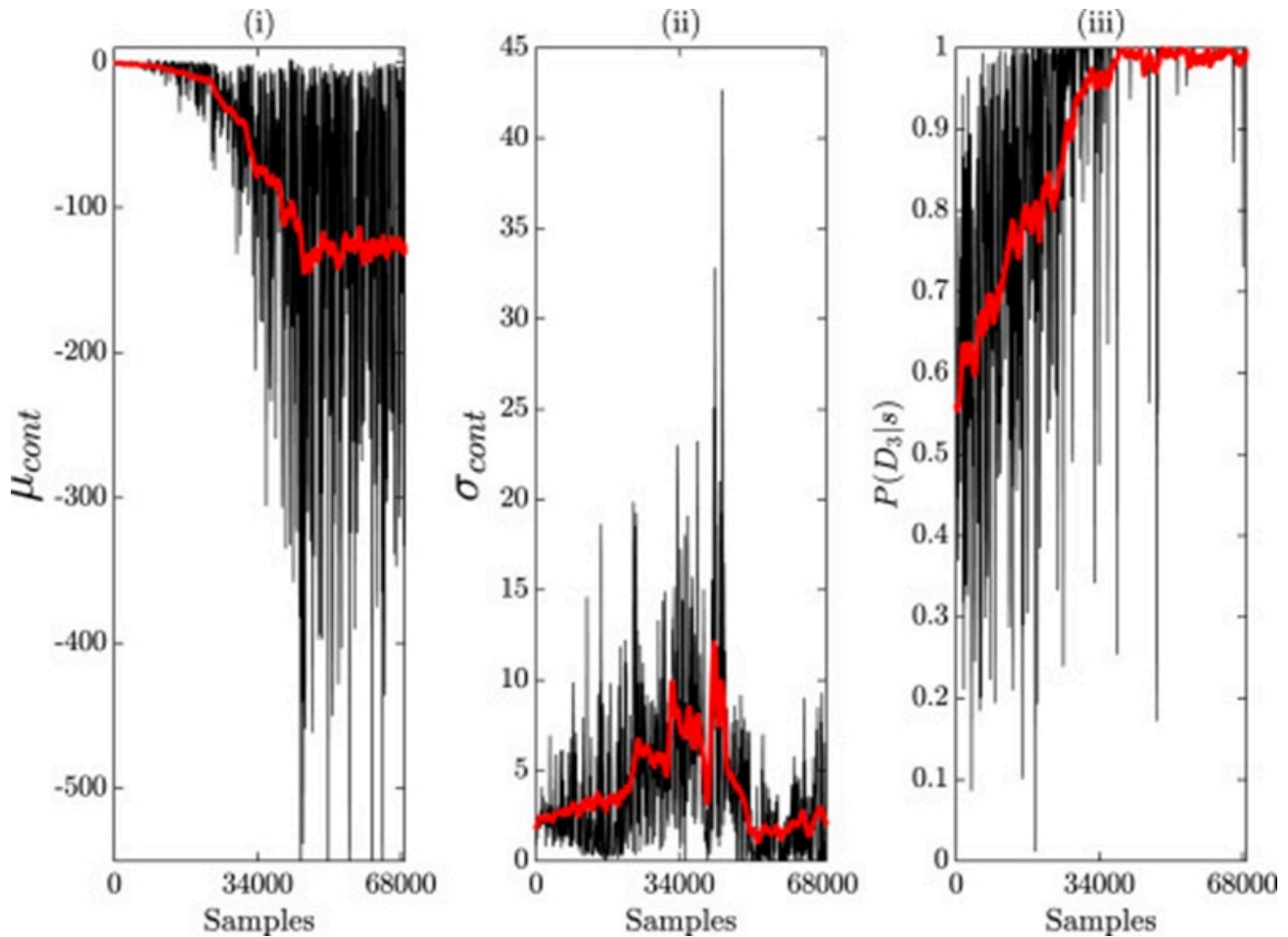


Figure 6. Evolution of policy parameters across episodes. The sub-figures illustrate (i) the mean output μ_{cont} , (ii) the standard deviation σ_{cont} , and (iii) the likelihood of executing the discrete action $P(D_3|s)$. To enhance clarity, only every 100th value is displayed (For details on the color references in this figure, see the web version of this article).

Test sets

The hybrid DRL controller was evaluated using three distinct test sets to assess its ability to handle both previously encountered and novel scenarios. The specifics of each test set are as follows:

- Test set 1: Generated using the same methodology as the training data, but applying a deterministic policy instead of the stochastic one used in training.
- Test set 2: Introduced previously unseen operating conditions (OCs) by expanding the variability in load and generation configurations. Instead of adjusting each load randomly within 80–120% , the range was extended to 70–130%.
- Test set 3: Simulated larger measurement uncertainties by multiplying each state variable by a random factor with a mean of 1 and a standard deviation of 0.01.

During training, the actor employed a stochastic policy, enabling it to explore the action space automatically. Although the exploration rate—reflected in the standard deviation of continuous actions and the stochasticity of discrete actions—declined toward the end of training, achieving a near-deterministic policy would require substantially longer training. For online deployment, it is preferable to convert the policy into a deterministic form, consistently selecting actions with the

highest probability of being optimal. Accordingly, during testing, continuous actions were determined by the mean μ_{cont} , while discrete actions were triggered whenever $P(Di|s) \geq 0.5$ for any defined Bernoulli probability.

Test results

Table 3 summarizes the hybrid DRL controller's performance over 200 episodes per test set. For test set 1, the average episode length was 1.09 steps, indicating that the controller generally maintained an adequate security margin in a single step. In test set 2, which involved new OCs with greater variability in initial load and generation, the controller continued to perform effectively, though the average episode length slightly increased to 1.12 steps, reflecting that some novel conditions required additional steps to restore the SOL. The mean total episode rewards were -28.6 for test set 1 and -26.3 for test set 2. For test set 3, which evaluated the effect of larger measurement errors, performance slightly declined: the average episode reward dropped to -34.7 , and the mean episode length rose to 1.19 steps, indicating that some scenarios required more steps to bring the SOL above the threshold. These results highlight the significance of including realistic random measurement errors in simulations, as such uncertainties are typically absent in purely simulated training data.

Table 3. Average performance of the hybrid DRL control.

Empty cell	Test set 1	Test set 2	Test set 3
Episode reward	-28.6	-26.3	-34.7
Episode length [steps]	1.09	1.12	1.19

Comparing hybrid and discrete-action controls

A key strength of the proposed hybrid DRL framework lies in its ability to adjust both continuous and discrete actions concurrently. To assess the significance of this capability, the hybrid DRL controller is compared with two alternative control strategies that operate exclusively within a discrete action space: one employing a rule-based lookup table, and the other using a DRL approach restricted to discrete actions. The details of these approaches and their associated actions are outlined below.

Rule-based lookup table control

Traditional preventive control methods typically require system operators to select actions by finding the closest match between the current system state and predefined states in a preventive control lookup table. Raw system measurements often require preprocessing, typically involving the computation of the security margin. If this margin falls below a specified threshold, operators take corrective measures. Using a rule-based lookup table, the procedure generally involves (i) estimating the security margin, (ii) executing actions to restore it, and (iii) re-evaluating the margin to ensure it exceeds the threshold. In contrast, the hybrid DRL controller can act directly on the system state without the time-consuming preprocessing step (i) or explicit calculation of the SOL.

Since the lookup table approach requires an initial SOL computation before any corrective action, a penalty is incorporated to fairly compare its performance with that of the hybrid DRL controller. The preprocessing and initial SOL calculation effectively add an extra episode step for the DRL controller, so a penalty of -50 (applied whenever the DRL agent fails to meet its control objective in a time step) and an additional episode step are included in the lookup table evaluation. However, for scenarios where the initial SOL exceeded the 30 MW threshold, no corrective actions were needed, and no penalty was applied. Based on the initial SOL, different actions (A1–A6) were triggered, as listed in **Table 4**, to control load curtailment and/or switch reactive shunts, corresponding to the continuous and discrete actions. Each action was designed to restore the SOL above the 30 MW threshold with a single intervention.

Table 4. Discrete actions for the discrete-DRL control and the rule-based lookup table.

Action	Initially	Actions activated	
Number	Estimated SOL	Load curtailment	Switching of
Empty cell	[MW]	[MW]	Reactive shunts
A1	SOL > 30	0	None
A2	0 < SOL ≤ 30	−50	D1
A3	−100 < SOL ≤ 0	−100	D1+D2
A4	−250 < SOL ≤ −100	−200	D1+D2+D3
A5	−350 < SOL ≤ −250	−350	D1+D2+D3
A6	SOL ≤ −350	−500	All

DRL control adapted for discrete actions

The DRL controller designed for a purely discrete action space is built using the same PPO algorithm as the hybrid controller, but modified to handle only discrete actions. In this setup, the DRL agent can select from the same six actions (A1–A6) listed in [Table 4](#) for the rule-based lookup control, but now chooses them directly without requiring an initial SOL estimate. The discrete DRL actor network maintains the same architecture as the hybrid controller, but its final layer uses a softmax activation instead of outputs for Normal and Bernoulli distributions. The softmax converts the outputs into a categorical probability distribution over the six discrete actions A1–A6, where each probability is proportional to the exponent of the corresponding input value:

$$S(z)_i = \frac{e^{z_i}}{\sum_{j=1}^6 e^{z_j}} \quad (12)$$

Actions can then be sampled from this distribution. Training used the same hyperparameters and number of episodes as the hybrid DRL controller, as specified in [Table 1](#), except for lower learning rates for the actor and critic networks $\alpha_{actor} = 1 \cdot 10^{-5}$ and $\alpha_{critic} = 5 \cdot 10^{-5}$ to prevent the discrete agent from prematurely converging to a sub-optimal policy.

Results

[Table 5](#) presents the total episode rewards and episode lengths for both the rule-based lookup table and the discrete DRL controller across all test sets, with the percentage difference relative to the hybrid DRL controller indicated in parentheses. Overall, the hybrid DRL controller outperformed the other methods on all test sets. For example, using the lookup table control increased the (negative) average reward from 109.0% in test set 3 to 153.7% in test set 1, whereas the discrete DRL control increased it from 14.7% to 25.2% across the same sets.

The episode length was notably higher for the lookup table method, largely because it requires two steps—estimating the initial SOL and confirming its restoration—whenever the SOL falls below 30 MW. In contrast, the episode lengths for the discrete DRL controller were comparable to those of the hybrid DRL controller. Despite similar time steps per episode, the hybrid DRL controller consistently achieved higher total rewards, highlighting that the ability to adjust both discrete and continuous actions allows for a more adaptive policy that more effectively regulates the power system's SOL. [Figure 7](#) shows histograms of total episode reward differences between (a) the hybrid DRL and lookup table controls and (b) the hybrid DRL and discrete DRL controls for scenarios in test set 1, revealing that the hybrid DRL outperformed the alternatives in 76.5% and 73.5% of scenarios, respectively.

Table 5. Average performance when using a rule-based lookup table control. Values in parentheses present the percentage increase in the average performance of the hybrid DRL control.

Test set	Rule-based lookup table Episode reward	Episode length [steps]	Discrete DRL control Episode reward	Episode length [steps]
Test set 1	−72.6 (153.7%)	1.77 (61.9%)	−35.9 (25.2%)	1.10 (1.4%)
Test set 2	−62.3 (136.4%)	1.65 (47.3%)	−33.3 (26.4%)	1.11 (−0.9%)
Test set 3	−72.6 (109.0%)	1.77 (47.7%)	−39.9 (14.7%)	1.16 (−3.3%)

It is important to emphasize that selecting a benchmark for comparison with the hybrid DRL control is not straightforward. While one might typically consider an optimization-based control approach, assessing the dynamic security margin (SOL) across multiple contingencies and choosing from a wide array of possible actions constitutes a highly nonlinear, nonconvex optimization problem. Addressing this with conventional optimization methods would either necessitate substantial model simplifications or demand excessive computational time, making it impractical for real-time applications by system operators.

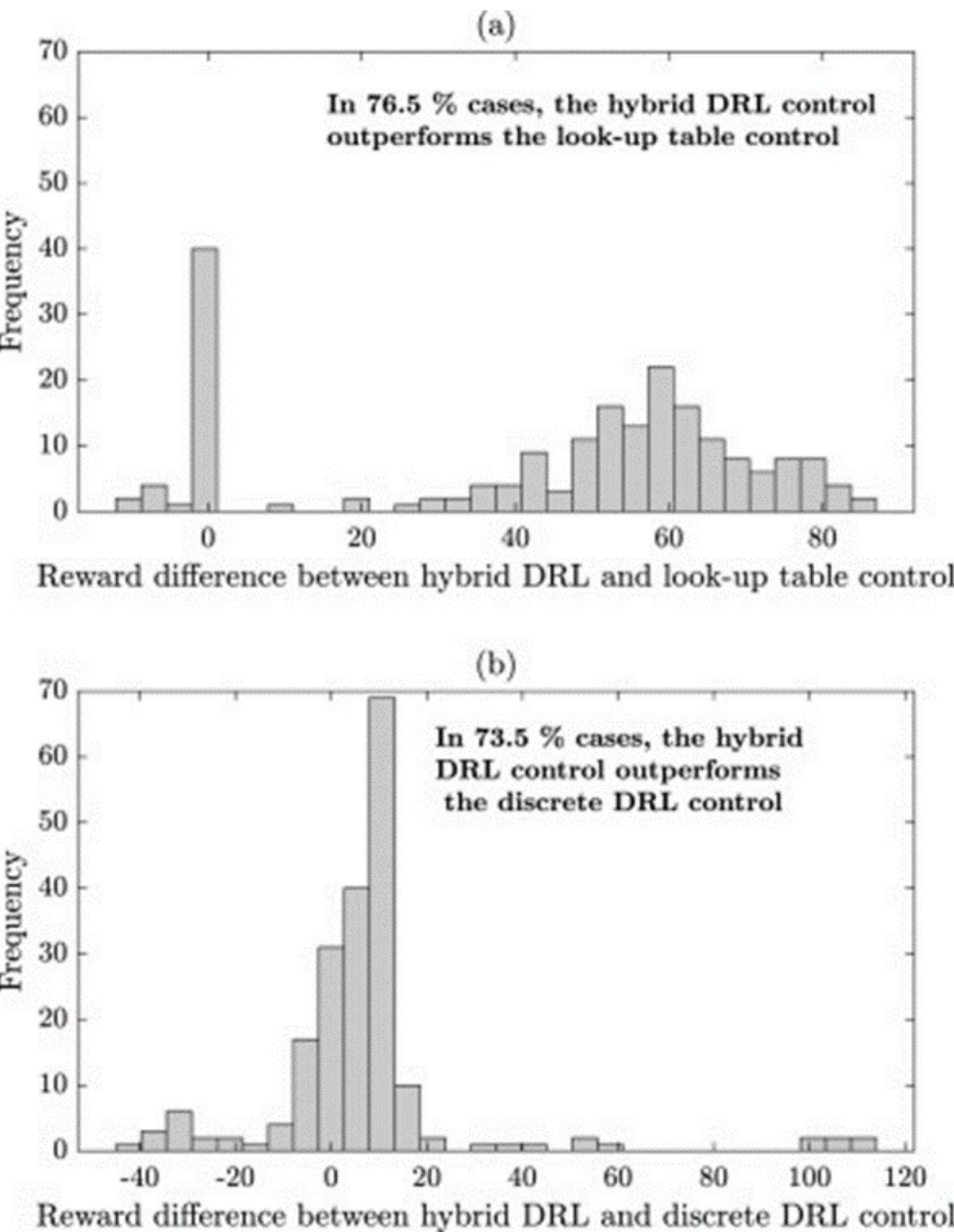


Figure 7. Histogram depicting the differences in episode rewards for test set 1: (a) between the hybrid DRL controller and the rule-based lookup table, and (b) between the hybrid DRL controller and the discrete DRL controller.

Conclusion

This work proposes a new real-time optimal control strategy designed to maintain the SOL above critical threshold levels. The approach leverages a DRL-based hybrid controller that can concurrently adjust discrete actions, such as switching shunt capacitors, and continuous actions, including generation rescheduling and load curtailment, to preserve system security. Evaluations on the constructed test sets showed that the hybrid controller successfully maintained the SOL above the thresholds in most scenarios at each time step. Its effectiveness decreased in scenarios with greater measurement inaccuracies, underscoring the need for comprehensive, representative training datasets. Comparisons with both a rule-based lookup table and a discrete DRL controller—each handling the same discrete action set—demonstrated that the hybrid DRL consistently delivered superior performance across all test cases.

Future work will focus on: (i) analyzing the effects of changes in network topology, (ii) broadening the stability assessment to cover additional phenomena like frequency stability, (iii) testing the controller's ability to generalize to unseen scenarios, and (iv) exploring performance and data requirements when applying the approach to larger-scale power systems.

Acknowledgements

None

Conflict of interest

None

Financial support

The authors declare the following financial interests/personal relationships, which may be considered as potential competing interests: Hannes Hagmar reports that Energiforsk AB provided financial support under project numbers EVU10450 and EVU10140. Hannes Hagmar also reports that Svenska Krafnät provided financial support.

Ethics statement

None

Received: 09 Mar 2025

Revised: 20 Apr 2025

Accepted: 03 Jun 2025

Published online: 20 September 2025

Rights and permissions

Open Access The author(s) retain copyright. This article is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. It may be shared and adapted for non-commercial purposes with appropriate attribution, an indication of changes, and distribution of adaptations under the same license. Third-party material may be subject to separate terms identified in its credit line. View the license at <https://creativecommons.org/licenses/by-nc-sa/4.0/>.

References

- Van Cutsem T, Moisse C, Mailhot R. Determination of secure operating limits with respect to voltage collapse. *IEEE Trans Power Syst.* 1999;14(1):327-35.
<https://doi.org/10.1109/59.744551..>
- Capitanescu F, Van Cutsem T. Preventive control of voltage security margins: A multicontingency sensitivity-based approach. *IEEE Trans Power Syst.* 2002;17(2):358-64.
<https://doi.org/10.1109/TPWRS.2002.1007901..>
- Zima M, Andersson G. Model predictive real-time control of electric power systems under emergency conditions. In: Savulescu S, ed. *Real-time stability in power systems*. Springer; 2004:367-85.
https://doi.org/10.1007/978-3319-06680-6_12.
- Cai H, Ma H, Hill DJ. A data-based learning and control method for long-term voltage stability. *IEEE Trans Power Syst.* 2020;35(4):3203-12.
<https://doi.org/10.1109/TPWRS.2020.2967434>.
- Li Q, Xu Y, Ren C. A hierarchical data-driven method for event-based load shedding against fault-induced delayed voltage recovery in power systems. *IEEE Trans Ind Inf.* 2021;17(1):699-709.
<https://doi.org/10.1109/TII.2020.2993807>.
- Ernst D, Glavic M, Wehenkel L. Power systems stability control: Reinforcement learning framework. *IEEE Trans Power Syst.* 2004;19(1):427-35.
<https://doi.org/10.1109/TPWRS.2003.821457>.
- Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, et al. Human-level control through deep reinforcement learning. *Nature.* 2015;518:529-33.
<https://doi.org/10.1038/nature14236>.
- Silver D, Schrittwieser J, Simonyan K, Antonoglou I, Huang A, Guez A, et al. Mastering the game of Go without human knowledge. *Nature.* 2017;550:354-9.
<https://doi.org/10.1038/nature24270>.
- Sallab AE, Abdou M, Perot E, Yogamani S. Deep reinforcement learning framework for autonomous driving. *Electron Imaging.* 2017;2017(19):70-6.
<https://doi.org/10.2352/ISSN.2470-1173.2017.19.AVM-023>.
- Gu S, Holly E, Lillicrap T, Levine S. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In: 2017 IEEE international conference on robotics and automation. 2017:3389-96.
<https://doi.org/10.1109/ICRA.2017.7989385>.
- Glavic M. (Deep) reinforcement learning for electric power system control and related problems: A short review and perspectives. *Annu Rev Control.* 2019;48:22-35.
<https://doi.org/10.1016/j.arcontrol.2019.09.008>.
- Vlachogiannis JG, Hatziaargyriou ND. Reinforcement learning for reactive power control. *IEEE Trans Power Syst.* 2004;19(3):1317-25.
<https://doi.org/10.1109/TPWRS.2004.831259>.
- Wang S, Duan J, Shi D, Xu C, Li H, Diao R, et al. A data-driven multi-agent autonomous voltage control framework using deep reinforcement learning. *IEEE Trans Power Syst.* 2020;35(6):4644-54.
<https://doi.org/10.1109/TPWRS.2020.2990179>.

Sun J, Zhu Z, Li H, Chai Y, Qi G, Wang H, et al. An integrated critic-actor neural network for reinforcement learning with application of DERs control in grid frequency regulation. *Int J Electr Power Energy Syst*. 2019;111:286-99.
<https://doi.org/10.1016/j.ijepes.2019.04.011>.

Huang Q, Huang R, Hao W, Tan J, Fan R, Cao Z. Adaptive power system emergency control using deep reinforcement learning. *IEEE Trans Smart Grid*. 2020;11(2):1171-82.
<https://doi.org/10.1109/TSG.2019.2933191>.

Glavic M. Design of a resistive brake controller for power system stability enhancement using reinforcement learning. *IEEE Trans Control Syst Technol*. 2005;13(5):743-51.
<https://doi.org/10.1109/TCST.2005.847339>.

Hashmy Y, Yu Z, Shi D, Weng Y. Wide-area measurement system-based low frequency oscillation damping control through reinforcement learning. *IEEE Trans Smart Grid*. 2020;11(6):5072-83.
<https://doi.org/10.1109/TSG.2020.3008364>.

Zarrabian S, Belkacemi R, Babalola AA. Reinforcement learning approach for congestion management and cascading failure prevention with experimental application. *Electr Power Syst Res*. 2016;141:179-90.
<https://doi.org/10.1016/j.epsr.2016.06.041>.

Zhou ZC, Wu Z, Jin T. Deep reinforcement learning framework for resilience enhancement of distribution systems under extreme weather events. *Int J Electr Power Energy Syst*. 2021;128:106676.
<https://doi.org/10.1016/j.ijepes.2020.106676>.

Dong Y, Xie X, Shi W, Zhou B, Jiang Q. Demand-response-based distributed preventive control to improve short-term voltage stability. *IEEE Trans Smart Grid*. 2018;9(5):4785-95.
<https://doi.org/10.1109/TSG.2017.2670618>.

Li Q, Yu Y, Lin T, Fu X, Du H, Wang Y, et al. Deep reinforcement learning-based fast prediction of strategies for security control. In: 2021 IEEE 5th conference on energy internet and energy system integration. 2021:2737-42.
<https://doi.org/10.1109/EI52483.2021.9713426>.

Neunert M, Abdolmaleki A, Wulfmeier M, Lampe T, Springenberg T, Romano F, et al. Continuous-discrete reinforcement learning for hybrid control in robotics. In: Conference on Robot Learning. PMLR; 2020:735-51.

Li H, Wan Z, He H. Real-time residential demand response. *IEEE Trans Smart Grid*. 2020;11(5):4144-54.
<https://doi.org/10.1109/TSG.2020.2978061>.

Qiu D, Wang Y, Zhang T, Sun M, Strbac G. Hybrid multi-agent reinforcement learning for electric vehicle resilience control towards a low-carbon transition. *IEEE Trans Ind Inf*. 2022;1-12.
<https://doi.org/10.1109/TII.2022.3166215>.

Silver D, Lever G, Heess N, Degris T, Wierstra D, Riedmiller M. Deterministic policy gradient algorithms. In: Xing EP, Jebara T, eds. Proceedings of the 31st international conference on machine learning. Vol 32. PMLR; 2014:387-95.

Sutton RS, Barto AG. Reinforcement learning: an introduction. MIT Press; 2018.

Williams RJ. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach Learn*. 1992;8(3-4):229-56.
<https://doi.org/10.1007/BF00992696>.

Tensorflow Keras Optimizers: RMSprop. https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/RMSprop. Accessed June 26, 2026.

Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347. 2017.

Van Cutsem T, Glavic M, Rosehart W, Canizares C, Kanatas M, Papadopoulos P, et al. Test systems for voltage stability studies. *IEEE Trans Power Syst*. 2020;35(5):4078-87.
<https://doi.org/10.1109/TPWRS.2020.2981491>.

Hagmar H, Tuan LA, Eriksson R. Impact of static and dynamic load models on security margin estimation methods. *Electr Power Syst Res*. 2022;202:107581.
<https://doi.org/10.1016/j.epsr.2021.107581>.

Van Cutsem T, Grenier M-E, Lefebvre D. Combined detailed and quasi steady-state time simulations for large-disturbance analysis. *Int J Electr Power Energy Syst*. 2006;28(9):634-42.
<https://doi.org/10.1016/j.ijepes.2006.03.005>.