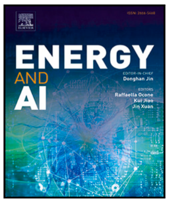
Real-time security margin control using deep reinforcement learning

Energy and AI 13 (2023) 100244

Contents lists available at ScienceDirect

Energy and AI

journal homepage: www.elsevier.com/locate/egyai

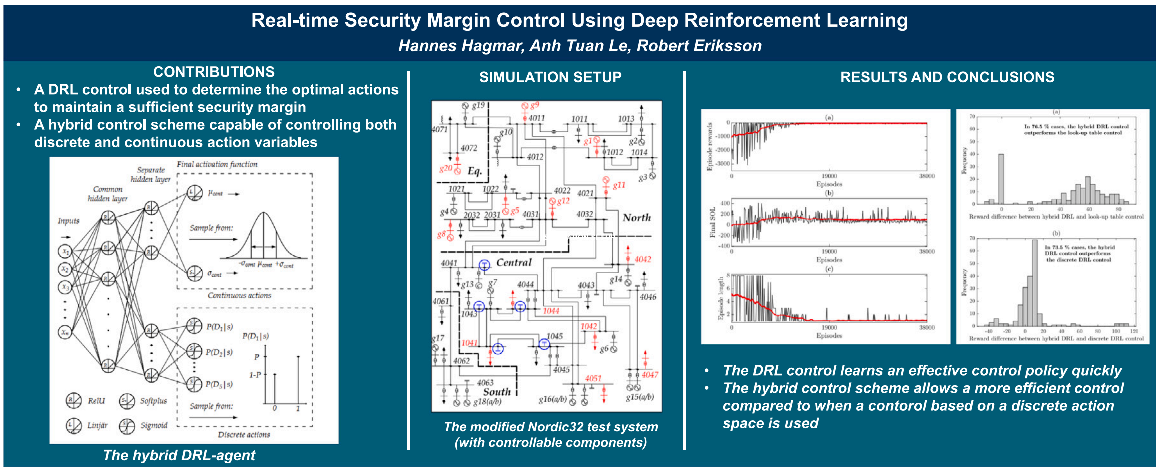


Real-time security margin control using deep reinforcement learning

Hannes Hagmar [a,*], Robert Eriksson [b], Le Anh Tuan [a]

[a] Chalmers University of Technology, Gothenburg 412 96, Sweden
[b] Svenska kraftnät (Swedish National Grid), Sundbyberg 172 24, Sweden

GRAPHICAL ABSTRACT



Real-time Security Margin Control Using Deep Reinforcement Learning

Hannes Hagmar, Anh Tuan Le, Robert Eriksson

CONTRIBUTIONS

- A DRL control used to determine the optimal actions to maintain a sufficient security margin
- A hybrid control scheme capable of controlling both discrete and continuous action variables

The hybrid DRL-agent

SIMULATION SETUP

The modified Nordic32 test system (with controllable components)

RESULTS AND CONCLUSIONS

- *The DRL control learns an effective control policy quickly*
- *The hybrid control scheme allows a more efficient control compared to when a contorol based on a discrete action space is used*

Common hidden layer
Separate hidden layer
Final activation function
Inputs
Sample from:
Continuous actions
Discrete actions
ReLU
Softplus
Linjär
Sigmoid
Eq.
North
Central
South
Episode rewards
Final SOL
Episode length
Episodes
In 76.5 % cases, the hybrid DRL control outperforms the look-up table control
In 78.5 % cases, the hybrid DRL control outperforms the discrete DRL control
Reward difference between hybrid DRL and look-up table control
Reward difference between hybrid DRL and discrete DRL control
Frequency

HIGHLIGHTS

- Deep reinforcement learning for real-time control of a dynamic security margin.
- Permitting real-time control in a highly complex and non-linear control problem.
- Allowing system operators to push their systems closer to the actual security limits.
- Evaluation of a continuous-discrete action space for a more flexible control policy.
- Providing an effective and low-system control of the secure operating limit.

ARTICLE INFO

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

ABSTRACT

This paper develops a real-time control method based on deep reinforcement learning aimed to determine the optimal control actions to maintain a sufficient secure operating limit. The secure operating limit refers to the limit to the most stressed pre-contingency operating point of an electric power system that can withstand a set of credible contingencies without violating stability criteria. The developed deep reinforcement learning method uses a hybrid control scheme that is capable of simultaneously adjusting both discrete and continuous action variables. The performance is evaluated on a modified version of the Nordic32 test system. The results show that the developed deep reinforcement learning method quickly learns an effective control policy to ensure a sufficient secure operating limit for a range of different system scenarios. The performance is also compared to a control based on a rule-based look-up table and a deep reinforcement learning control adapted

* Corresponding author.
E-mail address: hannes.hagmar@chalmers.se (H. Hagmar).

https://doi.org/10.1016/j.egyai.2023.100244
Received 3 October 2022; Received in revised form 23 January 2023; Accepted 19 February 2023
Available online 21 February 2023

for discrete action spaces. The hybrid deep reinforcement learning control managed to achieve significantly better on all of the defined test sets, indicating that the possibility of adjusting both discrete and continuous action variables resulted in a more flexible and efficient control policy.

1. Introduction

An electric power system that can withstand the loss of any single system component ($N-1$) without losing stability is generally referred to as being operated *securely*. To ensure that a power system is secure, system operators continuously estimate security margins and take preventive actions as soon as the security margins are deemed insufficient. This paper is devoted to the determination of the optimal control actions to ensure a sufficient *secure operating limit* (SOL). The SOL is the margin to the most stressed pre-contingency operating point that can withstand a set of credible contingencies without violating defined stability criteria [1]. The estimation of the SOL is computationally demanding and requires time-domain simulations (or a quasi-steady-state approximation) to account for the dynamic response after a disturbance. However, it generally provides a better measure of the security margin than conventional methods that are based on static assessments.

Typical preventive control actions used to ensure sufficient security margins include (1) management of reactive power resources and voltage control through tap changes and capacitor/reactor switching, and (2) generation rescheduling and load curtailment intended to relieve the loading of stressed transmission lines and buses [2]. Traditionally, power system stability control has been based on look-up tables that are pre-defined through offline simulations based on various typical scenarios. However, as control actions and their level of activation are correlated with different costs, such non-optimal control schemes can significantly decrease the operational efficiency of the system.

Optimal control methods have in several studies been proposed to deal with various types of power system stability control. However, power systems exhibit complex characteristics with a large number of states, nonlinear dynamics, and uncertainties [3]. To be able to compute the optimal control actions in a time frame required by system operators, simplifications of the system model or linear approximations are typically required [2]. Data-based control schemes, such as in [4,5], use an alternative approach based on an (offline) assessment of different states and actions. The optimal actions for each state are stored in a database and supervised learning algorithms are trained to map a state to a set of optimized actions. However, the complexity in assessing all the state–action pairs, especially if the number of available control actions is high, may result in slow learning and makes the method not suited to handle changes (e.g. in topology) of the underlying power system.

In recent years, significant progress has been made in solving complex control problems by using reinforcement learning (RL). RL is a data-driven approach where a control agent learns an optimal policy through interactions with a real power system or its simulation model [6]. Its combination with deep learning, called deep reinforcement learning (DRL), has proven effective in solving complex control problems in environments such as games [7,8], autonomous driving [9], and robotics [10]. DRL enables automatic high-dimensional feature extraction, making the control agent capable of handling a large number of states and actions that are involved in electric power system control. Previous implementations of RL in the field of power system stability control have so far mainly been focused on the control for systems operated in either a normal state or in an emergency state [11]. Implementations include methods adapted for automatic voltage control using either conventional RL [12] or DRL [13], frequency control [14], optimal load shedding [15], transient angle stability [16], and oscillation damping [17].

RL-based implementations adapted for preventive control purposes (e.g., to maintain a secure N-1 operation) found in the literature are relatively few. In [11], the authors argue that the reason may be because these control problems have traditionally been formulated as static optimization problems, which have typically been solved using more conventional optimization methods. However, a few studies have been conducted, including in [18], in which a RL-based implementation for preventive control is developed with the aim to determine the optimal control of active power generation for preventing cascading failures and blackouts. Another study aimed at resiliency enhancement was presented in [19], where a DRL framework for optimal rescheduling strategies in distribution systems was presented. In [20], a multi-objective RL scheme for generator set-points and compensation devices was developed to enhance short-term voltage security. A DRL-based method for predicting different strategies in security control was presented in [21], with a study focus on angle stability. To the best of our knowledge, there have been no previous implementations based on DRL where the aim has been to determine the optimal control actions to restore a sufficient security margin.

To achieve efficient preventive control, system operators are generally required to simultaneously control both discrete (e.g. switching of a shunt capacitor) and continuous (e.g. the level of active power generation rescheduling and load curtailment) action variables. However, state-of-the-art DRL algorithms such as deep Q-networks (DQN), or deep deterministic policy gradients (DDPG) are generally designed to only control *either* discrete *or* continuous action variables. A conventional approach is to transform all control variables so that they can be handled by a single control paradigm, for instance, by discretizing continuous variables or by approximating discrete actions as continuous by user-defined thresholds in the action space [22]. Although these approaches can work relatively well in practice in certain applications, they can also significantly impact the performance and make the control problem harder to solve. Previous studies have proposed adaptations of conventional DRL methods to extend them to handle a hybrid action space consisting of both discrete and continuous action spaces. In [23], a real-time demand response (DR) strategy was proposed where a control policy of both discrete and continuous actions was jointly optimized. In [24], a multiagent RL method was proposed to compute both discrete and continuous actions simultaneously to control various ancillary services provided by electric vehicles.

In this paper, we address the lack of preventive control methods and introduce a new method based on DRL that in real-time can determine the optimal control actions to maintain a sufficient SOL. Further, we also evaluate the limitations of different action spaces in DRL and propose a hybrid DRL control capable of adjusting both discrete and continuous action variables simultaneously. The main contributions can be summarized as:

- A DRL-based method that allows system operators to simultaneously monitor and control the advanced dynamic security margin SOL, which is not possible in real-time using conventional optimization methods. The control problem is adapted from a traditionally static optimization problem into a formulation that can be efficiently handled by the DRL framework. The DRL-based tool monitors the state through measurements, and if the security margin becomes too small, optimized control actions are suggested to system operators to steer the system back into a secure operation again. If the actions are not sufficient in restoring the security margin, the DRL-control reassesses the system state and new actions are taken until the system security is fully restored.
- A DRL architecture based on the proximal policy optimization (PPO) algorithm is developed that can handle a hybrid of continuous and discrete action spaces simultaneously. The hybrid control

is compared and evaluated with respect to more conventional control schemes based on a single action space. The possibility to use a hybrid action space ensures that the developed control policy is capable of controlling a range of different devices to make the system security margin restored efficiently.
- An evaluation into several aspects of the DRL control, including robustness to different simulation scenarios and noise in the inputs, is performed.

The remaining part of the paper is organized as follows. In Section 2, the theory regarding RL and the adaptations for the hybrid control are presented. In Section 3, the proposed method is presented with the steps for training the DRL agent. In Section 4, the results and the discussion are presented. Concluding remarks are presented in Section 5.

2. Reinforcement learning

Reinforcement learning is a data-driven approach used to solve complex and sequential optimal control problems. RL problems are generally modeled as discrete-time Markov decision processes (MDPs), where an agent uses its policy to interact with the MDP to give a trajectory of states, actions, and rewards. The received reward – also referred to as the reinforcement signal – is used to determine whether the taken actions were effective. The most common objective of the agent is to maximize the expected sum of future rewards over time. Through continuous interactions with the environment, the agent is then trained to achieve this goal [25].

In this paper, the considered MDP is comprised of: a state space $\mathcal{S}$; a hybrid action space with both discrete and continuous actions $\mathcal{A}$; an initial state distribution with density $p_1(s_1)$; a stationary transition dynamics distribution $p(s_{t+1}|s_t, a_t)$ which satisfies 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 $R : \mathcal{S}\times\mathcal{A} \to \mathbb{R}$. We denote a_t, s_t, and R_t the action, the state, and the reward, respectively, taken at time t [25]. A parametrized policy is used to select actions in the MDP. In the formulation used in this paper, the policy is assumed to be stochastic and can be formulated by $\pi_\theta(a_t|s_t) : \mathcal{S} \longrightarrow \mathcal{P}(\mathcal{A})$ where $\mathcal{P}(\mathcal{A})$ is a set of probability measures on $\mathcal{A}$, $\theta \in \mathbb{R}^n$ is a vector of n parameters, and $\pi_\theta(a_t|s_t)$ is a conditional probability density of taking action a_t in state s_t associated with the policy. The return G_t^γ is the total discounted reward from time step t and onward, defined as:

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

where $0 < \gamma < 1$ is a discounting factor. The value function is defined as the expected total discounted reward in state s when following the policy: $V^\pi(s) = \mathbb{E}\left[G_t^\gamma | s_t = s; \pi\right]$. The action-value function is defined as the expected total discounted reward in state s when taking action a and *then* following the policy: $Q^\pi(s, a) = \mathbb{E}\left[G_t^\gamma | s_t = s, a_t = a; \pi\right]$.

2.1. Policy gradients and actor–critic methods

Policy gradient methods are a class of model-free RL algorithms that learns a parametrized policy that can select actions without requiring a value function [26]. These methods seek to maximize a defined objective function $J(\theta)$ parametrized by θ, and their updates commonly approximate gradient ascent:

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

where $\widehat{\nabla J(\theta_t)}$ is a stochastic estimate of the gradient of the objective function with respect to θ_t [26] and α is the learning rate used in the optimization. One of the most commonly used gradient estimators in RL has the following form:

$$\widehat{\nabla J(\theta_t)} = \hat{\mathbb{E}}_t\left[\nabla_\theta \log \pi_\theta(a_t|s_t)\hat{A}_t\right] \tag{3}$$

where the expectation $\hat{\mathbb{E}}_t[\dots]$ indicates an empirical average over a batch of samples drawn from the MDP, $\nabla_\theta \log \pi_\theta(a_t|s_t)$ is the gradient of the parametrized policy, and $\hat{A}_t$ is an estimator of the advantage function at time step t, which can be formulated as:

$$\hat{A}_t = \hat{Q}^\pi(s_t, a_t) - \hat{V}_\phi^\pi(s_t) \tag{4}$$

where $\hat{V}_\phi$ and $\hat{Q}^\pi$ is an estimate of the value function and the action-value function, respectively. In the problem formulation used in this paper, relatively short episodic tasks are considered. This allows us to use the sample return G_t^γ from (1) to estimate the value of $\hat{Q}^\pi$, same as the approach used in the REINFORCE algorithm [27]. The value function is generally unknown and a function approximator is instead used, parametrized by a weight vector ϕ. The value function is learned simultaneously as the policy, commonly by minimizing a new cost function $L(\phi)$, based on the mean-squared error (or some other loss function) between the true value function $V^\pi(s_t)$ and its approximation $\hat{V}_\phi^\pi(s_t)$:

$$L(\phi) = \mathbb{E}_\pi\left[\left(V^\pi(s_t) - \hat{V}_\phi^\pi(s_t)\right)^2\right] \tag{5}$$

By computing the gradient of $L(\phi)$ and taking stochastic gradient-descent (or more efficient algorithms based on stochastic gradient-descent such as RMSprop [28]) on batches of data, the parameter vector ϕ that minimizes the loss can be found. By definition, G_t^γ is an unbiased estimate of the value function and can thus be used as a target value in the training [26]. The presented approach can be viewed as an actor–critic architecture where the policy π_θ is estimated by the *actor* and the value function $\hat{V}_\phi^\pi$ is estimated by the *critic*. The parameters used in forming the policy (θ) and the value function (ϕ) are in this paper representing the node weights of two separate neural networks, and the goal of training the networks is to find the optimal weights for these networks.

2.2. Proximal policy optimization

In this study, we use the Proximal Policy Optimization (PPO) algorithm, first presented in [29]. It is a policy gradient method that has a relatively simple implementation, higher sample efficiency than many other policy gradient methods, and is also well suited to adapt to handle a hybrid action space. In this paper, we use the "clipped" version of the PPO algorithm, with the objective function:

$$J^{clip}(\theta) = \hat{\mathbb{E}}_t\left[\min\left(r_t(\theta)\hat{A}_t, \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t\right)\right] \tag{6}$$

where r_t is a probability ratio given by:

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

and ϵ governs the clipping range of the objective function, and θ_{old} refers to the vector of policy parameters used in sampling the transitions and thus before any update of the policy parameters. The clipped objective function ensures that we do not move *too* far away from the current policy, which allows us to run multiple epochs of gradient ascent on the samples without causing destructively large policy updates. The r_t-ratio is always equal to 1 for the first epoch, when current policy $\pi_\theta(a_t|s_t)$ is the same as was used to sample the transitions $\pi_{\theta_{\text{old}}}(a_t|s_t)$. For each epoch, the policy is trained to *increase* the probability ratio r_t above 1.0 when the advantage function is *positive*, thus making advantageous actions more probable to be chosen by the policy. Similarly, the policy is trained to decrease the probability ratio r_t *below* 1.0 when the advantage function is *negative*, making disadvantageous actions less probable to be chosen by the policy in the future.

2.3. Adaptations for continuous-discrete control

Most state-of-the-art RL approaches have been optimized to work with either discrete or continuous action spaces [22]. A conventional approach is to transform all control variables so that they can be handled by a single control paradigm, for instance, by discretizing continuous variables or by approximating discrete actions as continuous by user-defined thresholds in the action space. Although these approaches can work relatively well in practice in certain applications, they can also significantly impact the performance and make the control problem harder to solve.

In this study, to handle a hybrid action space of both continuous and discrete actions, we follow the implementation introduced in [22], with a couple of adaptations. The hybrid policy $\pi_\theta(\mathbf{a}|s)$ is defined as a state-dependent distribution that jointly models discrete and continuous random variables. Independence between action dimensions, denoted by a^i, is assumed and the hybrid policy can be written as:

$$\pi_\theta(\mathbf{a}|s) = \pi^c_\theta(\mathbf{a}^{\mathcal{C}}|s)\pi^d_\theta(\mathbf{a}^{\mathcal{D}}|s) = \\ = \prod_{a^i\in\mathbf{a}^{\mathcal{C}}} \pi^c_\theta(a^i|s) \prod_{a^i\in\mathbf{a}^{\mathcal{D}}} \pi^d_\theta(a^i|s) \tag{8}$$

where $\mathbf{a}^{\mathcal{C}}$ and $\mathbf{a}^{\mathcal{D}}$ are subsets of action dimensions with continuous and discrete values respectively (where $\mathcal{C}$ and $\mathcal{D}$ represent continuous respectively discrete action spaces), and $\mathbf{a}$ is a vector of both discrete and continuous actions. We represent each component of the continuous policy π^c_θ as a normal distribution:

$$\pi^c_\theta(a^i|s) = \mathcal{N}\left(\mu_{i,\theta}(s), \sigma^2_{i,\theta}(s)\right) \tag{9}$$

where $\mu_{i,\theta}$ and $\sigma_{i,\theta}$ are the parametrized mean value and standard deviation of each continuous action dimension. We also represent each component of the discrete policy π^d_θ as a Bernoulli distribution parametrized by state-dependent probabilities $P_{i,\theta}(s)$:

$$\pi^d_\theta(a^i|s) = Bernoulli^i\left(P^i_\theta(s)\right) \tag{10}$$

where θ are the parameters of the policy components that we want to optimize. In this paper, we will represent the outputs $\mu_{i,\theta}(s), \sigma^2_{i,\theta}(s)$, $P_{i,\theta}(s)$ as outputs from a branched neural network. The continuous-discrete control and the used DRL architecture and its training are further discussed in Sections 3.4 and 3.5.

3. Proposed framework for real-time security margin control

In this section, the framework used in the real-time security margin control using DRL is presented. The DRL tool monitors the current state through measurements and can suggest optimized control actions to system operators whenever the SOL is below a certain threshold. The definitions used for the SOL are presented along with the used test system. The DRL agent's implementation details are presented, which include a thorough analysis of the neural networks, states, actions, rewards, and how the training data was generated.

3.1. Test system and security margin definitions

All simulations have been tested on the modified version of the Nordic32 test system, detailed in [30]. An overview of the system is presented in Fig. 1 . The SOL is defined as the limit to the most stressed state in which the system can still withstand a set of specified contingencies without violating defined stability criteria [1]. Thus, it provides a measure of how much further a system can be stressed and still be able to maintain $N-1$ security and provides a basis for the available transmission capacity of a power system. It has been shown to provide a more accurate measure of the security margin than other methods that are based on static assessments of the system, especially when the system is characterized by a larger share of loads with fast restoration dynamics [31]. To be able to account for the dynamic response after a disturbance, the SOL requires analysis using either dynamic simulations, approximations based on quasi-steady-state (QSS) [1], or combinations of the two [32].

System stress is typically defined as a combination of load demand increase and/or generation rescheduling, which are quantities that the system operator can observe and control in the pre-contingency state. A security margin is computed with respect to a set of credible contingencies. To reduce the number of simulations, it is important to perform contingency filtering to identify those contingencies that have low security margins for the current operating point. The SOL with respect to a chosen contingency can then be determined by a search process similar to the "binary search" proposed in [1]. The process is illustrated in Fig. 2 and is based on searching through a narrowing interval by iteratively testing the system security with respect to a dimensioning fault and different levels of system stress. The upper part in Fig. 2 illustrates the search process when the state was not secure, while the lower part in Fig. 2 illustrates the search process when the state was secure. Black dots indicate a secure state, while white dots indicate a state that is not secure. The search process is exemplified for a secure starting state. If the starting state is found secure, the system stress is increased by a total of ΔP. If the new state was found to be secure, the system stress was again increased with ΔP. In case it was found to be not secure, the system stress was instead reduced by $\Delta P/2$. The search process continued until a secure operating point was found and when the step size in system stress change was sufficiently small.

In the following simulations, the system stress is achieved by increasing the loads in the "Central" area of the modified Nordic32 test system, while simultaneously adjusting the generation in the "North" area with the same amount. The power factors of all loads were kept at their initial values and the distribution of the added load and generation were scaled by the initial load or the rated capacity of each generator. The dimensioning contingencies of concern will be the three largest generators that are located in the "Central" area, namely: generator *g14*, *g15*, and *g16*. The contingency resulting in the *lowest* SOL is always dimensioning for the system. An initial step size of $\Delta P = 128$ MW was used in the search process for the SOL and the process was stopped whenever the step size in system stress is equal to 1 MW.

In this study, the SOL was computed using dynamic simulations generated with PSS®E 35.0.0. A relatively simple stability criterion was used to determine whether a state was secure or not. The system was considered secure if, at the end of the post-contingency evaluation, *all* transmission bus voltages were above 0.90 pu. Although the modified Nordic32 test system is characterized by sensitivity towards long-term voltage instability, other types of instability can violate the defined stability criterion. For instance, transient angle instability can cause locally low voltages due to lost synchronism of certain generators. Frequency stability has not been included in the analysis but has been assumed to be stabilized by automatic frequency control actions of generators after a disturbance. All dynamic simulations ran for a maximum of 600 seconds but were stopped in advance if the case either collapsed (any bus voltage below 0.7 pu) or if the system stabilized early. This approach should ensure that the system had either stabilized or become unstable at the end of each simulation.

3.2. MDP formulation for security margin control

The SOL control problem is defined as an episodic MDP. At the beginning of each episode, the DRL agent receives a representation of the system state s_t through a set of measurements. Depending on the current policy π_θ, the DRL agent picks different actions which are activated in the system. The taken actions and the transition dynamics distribution cause the system state to change ($s_t \to s_{t+1}$) and give rise to a reward R_t. If the DRL agent managed to restore the SOL above the defined threshold, or if a maximum number of time steps have been reached, the episode ends. Otherwise, the DRL agent continues

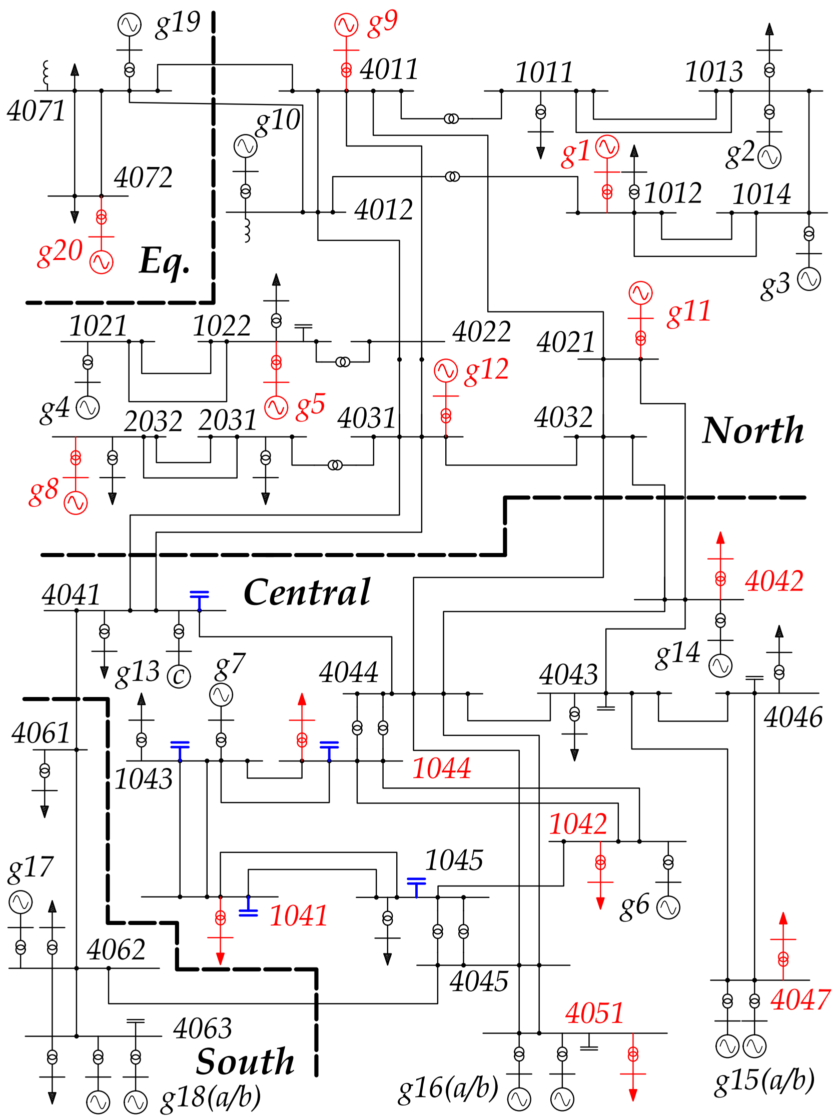


Fig. 1. One-line diagram of the modified Nordic32 system [30]. Loads and generators included in the continuous action space are marked in red, while shunt capacitors participating in the discrete action space are marked in blue. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

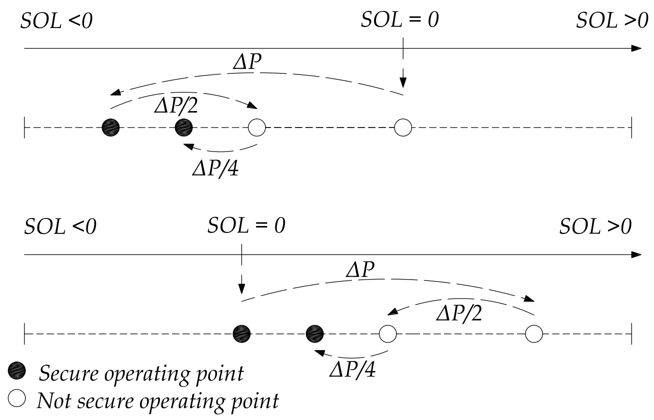


Fig. 2. Illustration of the search process for the SOL for a secure and a non-secure initial operating point.

observing new states, actions, and rewards. The MDP for the SOL control problem is defined as:

States: Ideally, the state used for the actor and the critic network should satisfy the Markov property and summarize all relevant information required to determine the optimal policy and value function at each time step. The problem formulation used in this paper is related to controlling a system in a pre-contingency steady state and therefore the static state (represented by the voltage magnitude and angles of all buses in the system) should be sufficient. When evaluating different choices of state parameters, it was found that using a state vector s_t consisting of measurements of the bus voltage magnitudes of all buses in the system and active and reactive power flow on all transmission lines and transformers, provided a more stable convergence than using voltage magnitudes and voltage angles. In addition, the current time step t of the episode was also added to the state vector. Neural networks are sensitive to input perturbations so to make the DRL agent more robust towards such errors, all state values were also randomly perturbed by multiplying each value with a random number sampled from a normal distribution with a mean of 1 and a standard deviation of 0.001. All states were then normalized by subtracting the mean of each state value and then dividing its standard deviation. The mean and standard deviation of each state value was computed from previously sampled states and a list with a maximum of 10 000 historic states was stored. Once 10 000 states were added to the list, the mean and standard deviation used for normalizing states became fixed.

Actions: the DRL agent can activate either continuous and/or discrete action variables. The continuous actions were used to reschedule power generation and perform load curtailment to reduce the transfer of power through the system. This was achieved by reducing the active load in the "Central" area at certain participating load buses: *4042*, *4047*, *4051*, *1041*, *1042*, *1044*, while simultaneously increasing the generation in the "North" area at certain participating generators: *G1*, *G5*, *G8*, *G9*, *G11*, *G12*, *G20*. The active load decrease was distributed on each of the participating load buses based on their initial load *before* the change, while the power factors of all loads were kept constant. The discrete action variables included the switching of an additional 100 MVAr of reactive power support from any, or several, shunt capacitors. The discrete actions ($D_1 - D_5$) controlled the shunt capacitors at the following buses: D_1: *1041*, D_2: *1043*, D_3: *1044*, D_4: *1045*, D_5: *4041*. The participating buses and equipment for the continuous and the discrete actions are all marked in red respectively blue, in the line diagram in Fig. 1.

State transition: The state transition dynamics are deterministic and governed by a set of differential and algebraic equations used to build the dynamic model in PSS®E.

Rewards: The reward R_t for the taken actions was computed by a combination of the resulting SOL and the costs for the continuous (C_{cont}) and the discrete (C_{disc}) actions. In this study, the reward is unitless, but should in real applications reflect the actual monetary cost of different actions and the corresponding rewards of the control goal. *Any* activation of the discrete actions contributed to a negative reward of −5, representing the cost of the mechanical wear that is involved in switching the shunt capacitors. The cost for the continuous actions contributed to a negative reward of −0.1 per adjusted MW in the power transfer. Changing a total of ± 200 MW would thus result in a negative contribution to the reward of −20. This negative reward reflects the system cost of market adjustments or load/generation curtailment. The control goal is to always restore the SOL to a value equal to or above 30 MW, which would ensure that a sufficient security margin is achieved and that the $N-1$ contingency criterion always would be satisfied with some margin to account for possible inaccuracies. If the SOL was below 30 MW, a negative reward of −50 was added to the total reward for that time step. The negative rewards and the costs for the actions are chosen so that the DRL agent should always strive to control the system to achieve a sufficient SOL in as few steps as possible, while still minimizing the actions activated. The final reward when accounting for the resulting SOL was then computed as:

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

3.3. Training data generation

An overview of the steps involved in training data generation and the training of the DRL agent is illustrated in Fig. 3. The different steps are detailed in the sections below. To speed up training, the data generation was parallelized and multiple CPU cores were used to generate data.

(1) *Generate initial operating condition*: A large range of different initial OCs was generated to serve as training data for the algorithm. All loads in the system were randomly and individually varied by multiplying the active load value with a random variable generated from a uniform distribution (using 90 % of the original load as a lower limit and 130% of the original load as an upper limit). The power factors of all loads were kept constant. The total change in loading was distributed proportionally among all the generators in the system based on the initial active power produced by each generator. The generated initial OCs were solved by a full Newton–Raphson load flow and were re-initialized in case the load flow did not converge.

Table 1
Design and hyperparameters used in training.

		Parameter	Values
Architecture	*Critic*	Number of inputs	499
		Neurons in first layer	128
		Neurons in second hidden layer	64
		Final activation function	Linear
		Hidden layer activation	RelU
	Actor	Number of inputs	499
		Neurons in 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 $P(D_x\|s)$	Sigmoid
		Hidden layer activation	RelU
	Training	Max Epochs (K)	10
		PPO clip parameter (ϵ)	0.2
		Discount factor (γ)	0.99
		Optimizer	RMSprop [28]

(2) *Sample state s_t and actions a_t*: Once an initial OC was generated, the state s_t was sampled from the system and passed to the actor network. The actor network outputs parameters that form the current hybrid policy $\pi_\theta(\mathbf{a}|s)$ from which a set of actions were sampled. The actor network and how it is used to form the hybrid policy is further detailed and discussed in Section 3.4.

(3) *Take actions and solve load flow*: Once the actions were sampled and activated in the system, a new Newton–Raphson load flow was computed which formed the state transition from $s_t \to s_{t+1}$.

(4) *Compute the SOL*: After the load flow for the new state was solved, the SOL for the new operating point was computed following the steps defined in Section 3.1.

(5) *End episode and save transitions*: The episode was ended if either the SOL was restored above 30 MW (the defined security margin for when the system was assumed to be secure) or if the current time step was equal to $T = 8$. At the end of all episodes, the transition data gathered during the episode (s_t, a_t, R_t, …, s_T, a_T, R_T) was stored and later used during training. The training data generation was reiterated for a total of $N = 64$ episodes before it was trained on the generated data.

3.4. Architecture of actor and critic network

The actor network is illustrated in Fig. 4 and further detailed in Table 1. It shares a common hidden layer, then a separate hidden layer is used for each type of activation function. The network outputs parameters used in defining the Normal distribution (9) and the Bernoulli distributions (10) that are used for the continuous and discrete action variables, respectively. The Normal distribution is parametrized by a mean value μ_{cont} and a standard deviation σ_{cont}, while the Bernoulli distribution is parametrized by a single probability parameter ranging from $0-1$. The mean value μ_{cont} is computed using a linear activation function in the final layer. The standard deviation σ_{cont} is computed using a Softplus activation function that ensures that the value never becomes negative. Finally, the Bernoulli distribution is achieved using a sigmoid activation function in the final layer that ensures that the output is bounded between 0 and 1. The critic network is separate from the actor network and consists of a simple NN with two hidden layers and a linear final activation function, further detailed in Table 1.

3.5. Training actor and critic networks

Once a batch of training data was sampled, the actor and critic networks were trained. The training was performed using the software Tensorflow in Python which automatically computes the gradients on the defined cost functions. The critic network was first used to estimate

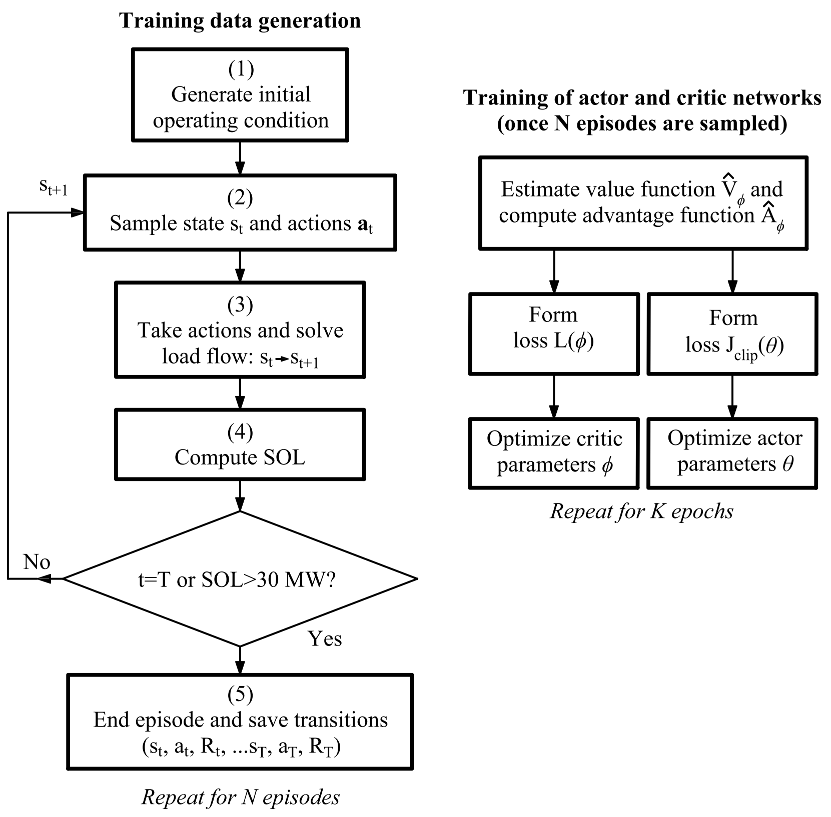

Training data generation
(1) Generate initial operating condition
s_{t+1}
(2) Sample state s_t and actions $\mathbf{a}_t$
(3) Take actions and solve load flow: $s_t \rightarrow s_{t+1}$
(4) Compute SOL
t=T or SOL>30 MW?
No
Yes
(5) End episode and save transitions (s_t, a_t, R_t, ...s_T, a_T, R_T)
Repeat for N episodes
Training of actor and critic networks (once N episodes are sampled)
Estimate value function $\hat{V}_\phi$ and compute advantage function $\hat{A}_\phi$
Form loss $L(\phi)$
Form loss $J_{clip}(\theta)$
Optimize critic parameters ϕ
Optimize actor parameters θ
Repeat for K epochs

Fig. 3. Flowchart showing the generation of training data and the training of the actor and critic network.

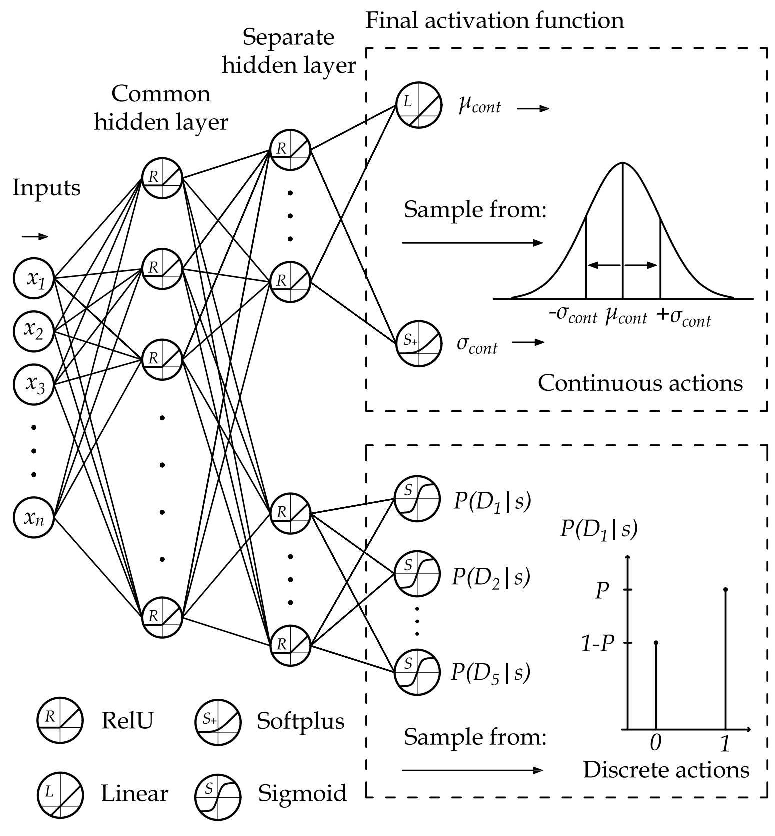


Fig. 4. Architecture of the actor network showing the outputs forming the stochastic continuous-discrete policy.

Table 2
Learning rates over training iterations.

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

the value function $\hat{V}^{\pi}_{\phi}$ of each state. The returns G^{γ}_{t} were estimated using (1). The advantage estimates $\hat{A}_t$ were estimated using (4) and the estimated value of the state $\hat{V}^{\pi}_{\phi}$ given by the critic network. A γ of 0.99 was used to compute the returns. The final objective function for the actor network was computed by taking the mean value of all samples for all N episodes on J^{clip}. The critic objective function was defined as the mean squared error (MSE) of the value function error, computed from (5). The final critic objective function was formed by taking the mean value of all samples for all N episodes on the MSE of the value function error.

Once the actor and critic objectives were formed, they were optimized using the RMSprop algorithm, which is an adaptable algorithm suitable for gradient-based optimization of stochastic objective functions. It should be noted that different optimization algorithms and hyperparameters were evaluated during the study design, but only the settings that provided the best results are presented in Table 1. Dropout, a regularization technique where a certain percentage of the connections between each layer of neurons are randomly masked (or "dropped") to reduce overfitting, was tested but was found to not improve the performance. The actor objective function was then maximized with respect to θ, while the critic network objective function was minimized with respect to ϕ. The training was performed for $K = 10$ epochs on the whole batch of N episodes simultaneously. It should be mentioned that although a search for suitable hyperparameters was conducted, the performance could have been improved even more by further optimizing training parameters such as the learning rate or the number of hidden neurons in each layer. To speed up training and stabilize it during later stages, the learning rate was adjusted as the number of training iterations increased and is specified in Table 2.

4. Results and discussion

4.1. Training results

The hybrid DRL agent was trained for a total of 600 training iterations, corresponding to 38,400 different episodes and a total of 68,900 samples (each episode consisted of up to 8 time steps/samples, depending on the episode length). The episode reward varied relatively significantly between different episodes. To avoid terminating the training prematurely, the average episode reward was computed for every 100 training episodes. If the average reward did not improve compared to the previous 100 training episodes, the training was assumed to have converged and was then stopped.

The training performance is presented in Fig. 5. The total episode reward is presented in sub-figure (i), the final SOL of the episode is presented in sub-figure (ii), and the number of episode time steps is presented in sub-figure (iii). The red line shows a centered moving average computed over the mean value of 500 episodes. To better visualize the results, only every *100th* value during the training is illustrated in the figure. The results show that the performance improved rapidly until around 19,000 episodes, after which the policy managed to achieve a SOL above the threshold value of 30 MW using only a single time step for a majority of the episodes. After this, the performance improved by mainly optimizing the level of action activation for each scenario.

In Fig. 6, the development of the different policy parameters is presented. The policy parameter governing the standard deviation σ_{cont} increased at first, which was then followed by the mean value μ_{cont} being adjusted. After the model was trained on approximately 40,000 samples, the mean value μ_{cont} stabilized. After that, the model improved mainly by reducing its exploration rate (the standard deviation σ_{cont} and the randomness of the discrete actions). In sub-figure (iii) of Fig. 6, the probability of the taken action D_3 is illustrated, showing that the policy became more and more certain of whether the discrete action should be activated or kept inactivated. Similar training development for the other discrete actions was observed as well.

4.2. Test sets

The hybrid DRL control was tested on three different test sets to evaluate its performance to handle different types of seen and unseen scenarios. Each of the test sets is detailed below.

1. **Test set 1**: Data generated in the same way as for the training data, but using a deterministic policy instead.
2. **Test set 2**: Introducing new unseen OCs by increasing the variation of the generation and load configurations. Instead of randomly adjusting each load between 80% to 120% as specified in Section 3.3, the OCs were adjusted randomly between 70% to 130%.
3. **Test set 3**: Introducing larger measurement errors by multiplying each state value with a random number with a mean of 1 and a standard deviation of 0.01.

During training, the actor used a stochastic policy which allowed it to automatically explore the available action space. While the exploration rate (the standard deviation of the continuous action space and the randomness of the discrete action spaces) decreased during the final part of the training, it would require a significantly longer training time for it to reach a point where it essentially converged towards a *fully* deterministic policy. When implementing the policy online it is then more suitable to transform the control policy into a deterministic one and always pick the actions that with the *highest probability* are optimal. When testing the algorithm, the continuous action was thus controlled directly by the mean value μ_{cont}. Each of the discrete actions was activated whenever any of the defined Bernoulli probabilities satisfied $P(D_i|s) \geq 0.5$.

4.3. Test results

The performance of the hybrid DRL control when tested on the different test sets is presented in Table 3, each consisting of 200 episodes. For test set 1, the average episode length was 1.09 steps, indicating that the hybrid DRL control managed to ensure a sufficient security margin in a single time step for a majority of all scenarios. For test set 2, where new unseen OCs were introduced by increasing the variation by which the loads and generation were initialized, showed good performance as well. The average episode length increased slightly to 1.12 steps, indicating that some of the unseen OCs forced the hybrid DRL control to require a few more steps before the SOL was restored. The average total episode reward for each of these test sets was −28.6 and −26.3, respectively. For test set 3, where the impact of larger measurement errors on the state values was evaluated, the performance dropped slightly. The average episode reward was reduced to −34.7, while the average episode length increased to 1.19 steps. Thus, for several of the scenarios in the test set, the hybrid DRL control required slightly more time steps before the SOL was restored above the threshold. The results also indicate the importance of incorporating random errors that exist in real power systems, but which are generally not present when training the method on purely simulated data.

4.4. Comparing controls based on hybrid and discrete action spaces

The main advantage of the proposed hybrid DRL architecture is the capability to simultaneously adjust *both* discrete and continuous

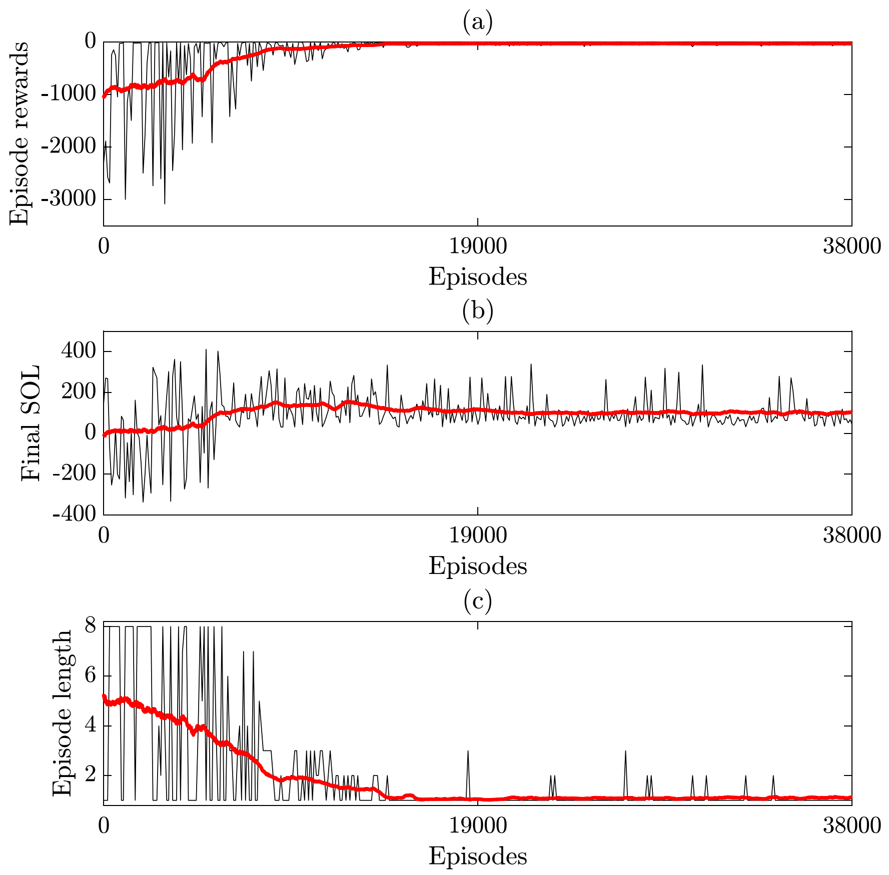


Fig. 5. Performance and development over training samples and episodes: Sub-figures showing (i) episode rewards during all episodes; (ii) final SOL; (iii) episode length. The red line indicates a moving average computed over the mean of 500 episodes. For better visualization, every 100th value is illustrated. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

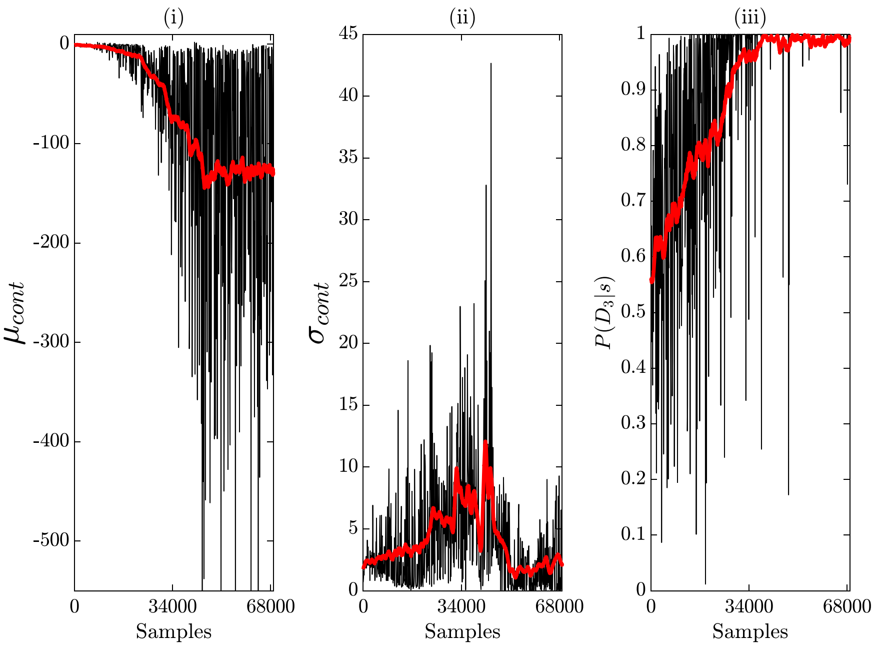


Fig. 6. Development of policy parameters over episodes. Sub-figures showing (i) the mean value output μ_{cont}; (ii) the standard deviation output σ_{cont}; (iii) the probability of taking the discrete action $P(D_3|s)$. For better visualization, every 100th value is illustrated. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

action variables. To evaluate the impact of this feature, the developed hybrid control is compared to two other control policies which are only capable of controlling a discrete action space: one which is based on a rule-based look-up table control, while the other is based on DRL but

Table 3
Average performance of the hybrid DRL control.

	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

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

Action	Initially	Actions activated	
number	estimated SOL [MW]	Load curtailment [MW]	Switching of reactive shunts
A1	SOL > 30	0	None
A2	$0 < \text{SOL} \leq 30$	−50	D_1
A3	$-100 < \text{SOL} \leq 0$	−100	$D_1 + D_2$
A4	$-250 < \text{SOL} \leq -100$	−200	$D_1 + D_2 + D_3$
A5	$-350 < \text{SOL} \leq -250$	−350	$D_1 + D_2 + D_3$
A6	$\text{SOL} \leq -350$	−500	All

only adapted for discrete action spaces. Each of the two methods and their actions are discussed in the following subsections.

4.4.1. Rule-based look-up table control

Conventional methods for preventive control typically rely on system operators to choose actions by matching the current system state with the nearest system state defined in a preventive control look-up table. The difficulty of assessing raw measurements from the system, require system operators to pre-process the system state, generally by first computing the security margin and then, in case the security margin is below a defined threshold, taking measures to restore it. Thus, using a control based on a rule-based look-up table, a system operator would have to (i) first estimate the security margin, (ii) possibly activate actions to restore it, and finally (iii) re-evaluate the security margin to ensure that it is above the defined threshold. In comparison, the developed hybrid DRL control can take actions by *directly* monitoring the system state, without the need for the additional first step (i) with time-consuming data pre-processing and computation of the SOL.

Since the rule-based look-up table require an initial computation of the SOL before any actions can be initiated, a penalty is added to be able to compare its performance to that of the hybrid DRL control. The pre-processing of measurement data and the initial computation of the SOL corresponds to an additional episode step for the hybrid DRL control. Thus, a penalty of −50 (the penalty added for the DRL agent at every time step it does not achieve its control goal) and an added time step for each episode, are added to the performance of the rule-based look-up table. However, for the scenarios when the initially estimated SOL was *above* the threshold value of 30 MW, no actions had to be taken and thus no penalty was added for those scenarios. Depending on the initially estimated SOL, different actions (A1-A6) were activated, each specified in Table 4. The actions activated controlled load curtailment and/or switching of reactive shunts which refer to the continuous respectively discrete actions specified in Section 3.2. The actions and their SOL activation levels were designed to *always* ensure that the SOL was restored above the threshold of 30 MW using a single action.

4.4.2. DRL control adapted for discrete actions

The DRL control that only handles a discrete action space is developed using the same PPO algorithm as for the hybrid control but is adapted only for a discrete action space. The discrete DRL control can choose from the *same* actions (A1–A6) defined in Table 4 as for the rule-based look-up control, but each of the actions is now instead chosen directly by the DRL agent without the need of an initially estimated SOL. The discrete DRL actor network has an identical architecture to the one used for the hybrid control, but instead of forming outputs used in defining the Normal and Bernoulli distributions, it uses a softmax activation function in the final layer. The used softmax function normalizes the outputs into a categorical probability distribution consisting of six numbers (the available discrete actions A1–A6) where each probability is proportional to the exponents of the input numbers:

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

for $i = 1, \ldots, 6$ and $\boldsymbol{z} = (z_1, \ldots, z_6) \in \mathbb{R}^6$, where 6 is the number of available discrete actions. From the defined categorical probability distribution, different discrete actions could then be sampled. The same hyperparameters and number of training episodes as was used for the hybrid DRL control and as is defined in Table 1 was used during training, with the exception that a lower learning rate for the actor and critic network of $\alpha_{actor} = 1 \cdot 10^{-5}$ and $\alpha_{critic} = 5 \cdot 10^{-5}$ was used, respectively. The lower learning rates were used to ensure that the discrete DRL agent did not converge to a sub-optimal policy due to a too high initial learning rate.

4.4.3. Results

In Table 5, the total episode reward and the episode length for the rule-based look-up table and the discrete DRL control are presented for each of the defined test sets. For each metric and test set, the percentage difference in performance compared to the hybrid DRL control is also presented in parenthesis after each value. The results show that the proposed hybrid DRL control performed significantly better on all of the defined test sets. For instance, the (negative) average reward increased from 109.0% for test set 3, to 153.7% for test set 1 when the rule-based look-up table control was used. In the case of the discrete DRL control, the (negative) average reward increased from 14.7% for test set 3, to 25.2% for test set 1.

The episode length for the rule-based look-up table was significantly higher for all of the test sets, mostly caused by the requirement of the look-up table control to take two steps (one to estimate the initial SOL, and one to verify that the SOL had been restored) whenever the initial SOL was below the threshold value of 30 MW. The average episode length for the discrete DRL control was in a similar range as for the hybrid DRL control for the different test sets. However, although both the hybrid DRL control and the discrete DRL control required a similar number of time steps in each episode, the hybrid DRL control achieved a significantly better total episode reward on each test set. This indicates that the possibility of both adjusting discrete and continuous action variables results in a more flexible control policy that more efficiently can adjust the SOL of a power system. In Fig. 7, a histogram showing the total episode reward difference between (a) the hybrid DRL control and the look-up table control, and (b) the hybrid DRL control and the discrete DRL control, is presented for the different scenarios included in test set 1. The results show that in 76.5% and 73.5% of all scenarios respectively, the hybrid DRL control achieved a better performance than the other types of control.

Finally, it should be stressed that the choice of which method to compare the hybrid DRL control to is not trivial. A typical choice would be to evaluate it against some optimization-based control method. However, evaluating a dynamic security margin (the SOL) with respect to a number of different contingencies and choosing from a wide range of different actions is a highly non-linear and non-convex optimization problem. Solving such a problem with optimization-based methods would either require significant simplifications in the model or would be too time-consuming to achieve in the time frame required by system operators, making the comparison impractical.

5. Conclusion

This paper introduces a new method for optimal control to maintain a sufficient SOL in real-time. The optimal control method is based on DRL and introduces a hybrid control scheme that can simultaneously

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

	Rule-based look-up table		*Discrete DRL control*	
	Episode reward	Episode length [steps]	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%)

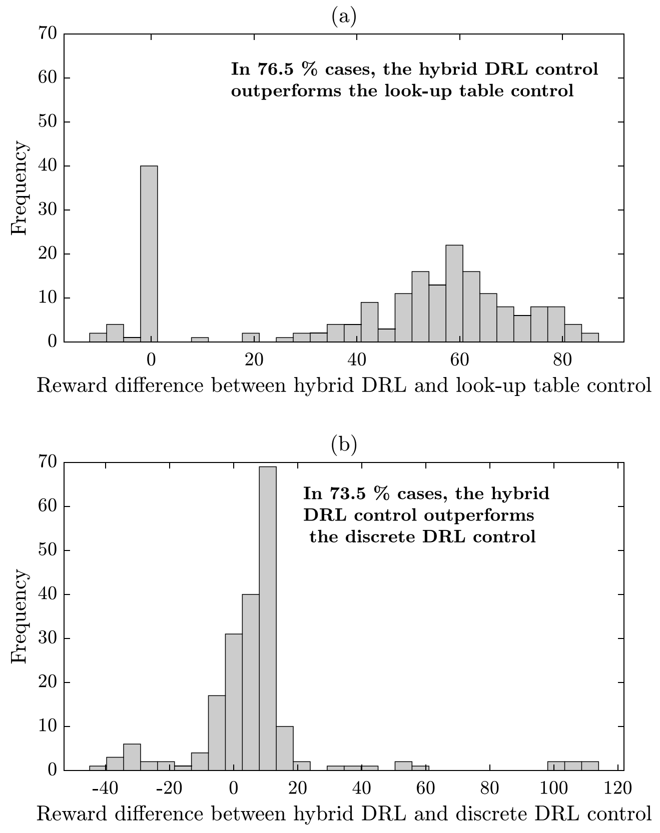


Fig. 7. Histogram showing the episode reward difference between (a) the hybrid DRL control and the look-up table control and (b) the hybrid DRL control and the discrete DRL control, given for test set 1.

control both discrete (switching of a shunt capacitor) and continuous (generation rescheduling and load curtailment) action variables to ensure that the SOL is above defined threshold values. The method showed good performance on the developed test sets and managed to control the SOL above the defined threshold values for a single time step for a majority of the scenarios. When tested on disturbance scenarios that included larger measurement errors, the method's performance dropped, which highlights the need for representative training data. The method was further compared to the performance of a rule-based look-up table and a discrete DRL control, which both controlled the same number of discrete actions. The results showed that the control of the hybrid DRL agent achieved significantly better on all of the defined test sets.

Future research work includes (i) extending the study to evaluate the impact of various topology changes; (ii) adapting and expanding the stability criteria to also include other stability phenomena such as frequency stability; (iii) further evaluating the generalization capability of DRL control to handle scenarios not included in the training; (iv) evaluate the performance and the impact on training data requirements when adapting the method for larger systems.

Declaration of competing interest

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

Data availability

The authors are unable or have chosen not to specify which data has been used.

References

[1] 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. http://dx.doi.org/10.1109/59.744551.
[2] 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.
[3] Zima M, Andersson G. Model predictive real-time control of electric power systems under emergency conditions. In: Savulescu S, editor. Real-time stability in power systems. Springer; 2004, p. 367–85. http://dx.doi.org/10.1007/978-3-319-06680-6_12.
[4] 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. http://dx.doi.org/10.1109/TPWRS.2020.2967434.
[5] 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. http://dx.doi.org/10.1109/TII.2020.2993807.
[6] Ernst D, Glavic M, Wehenkel L. Power systems stability control: Reinforcement learning framework. IEEE Trans Power Syst 2004;19(1):427–35. http://dx.doi.org/10.1109/TPWRS.2003.821457.
[7] Mnih V, Kavukcuoglu K, Silver D, Rusu D, Veness AA, et al. Human-level control through deep reinforcement learning. Nature 2015;518:529–33. http://dx.doi.org/10.1038/nature14236.
[8] Silver D, Schrittwieser J, Simonyan K, Antonoglou I, Huang A, et al. Mastering the game of go without human knowledge. Nature 2017;550:354–9. http://dx.doi.org/10.1038/nature24270.
[9] Sallab AE, Abdou M, Perot E, Yogamani S. Deep reinforcement learning framework for autonomous driving. Electron Imaging 2017;2017(19):70–6.
[10] 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, p. 3389–96. http://dx.doi.org/10.1109/ICRA.2017.7989385.
[11] 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. http://dx.doi.org/10.1016/j.arcontrol.2019.09.008.
[12] Vlachogiannis JG, Hatziargyriou ND. Reinforcement learning for reactive power control. IEEE Trans Power Syst 2004;19(3):1317–25. http://dx.doi.org/10.1109/TPWRS.2004.831259, cited by: 83.
[13] Wang S, Duan J, Shi D, Xu C, Li H, et al. A data-driven multi-agent autonomous voltage control framework using deep reinforcement learning. IEEE Trans Power Syst 2020;35(6):4644–54. http://dx.doi.org/10.1109/TPWRS.2020.2990179.
[14] Sun J, Zhu Z, Li H, Chai Y, Qi G, 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. http://dx.doi.org/10.1016/j.ijepes.2019.04.011.
[15] Huang Q, Huang R, Hao W, Tan J, Fan R, et al. Adaptive power system emergency control using deep reinforcement learning. IEEE Trans Smart Grid 2020;11(2):1171–82. http://dx.doi.org/10.1109/TSG.2019.2933191.

[16] 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. http://dx.doi.org/10.1109/TCST.2005.847339, cited by: 28; All Open Access, Green Open Access https://www.scopus.com/inward/record.uri?eid=2-s2.0-26244451387&doi=10.1109%2fTCST.2005.847339&partnerID=40&md5=e49eebd30d08b62fd80f11f48811de64.

[17] 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. http://dx.doi.org/10.1109/TSG.2020.3008364.

[18] 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. http://dx.doi.org/10.1016/j.epsr.2016.06.041.

[19] chen Zhou Z, 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. http://dx.doi.org/10.1016/j.ijepes.2020.106676.

[20] 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. http://dx.doi.org/10.1109/TSG.2017.2670618.

[21] Li Q, Yu Y, Lin T, Fu X, Du H, 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, p. 2737–42. http://dx.doi.org/10.1109/EI252483.2021.9713426.

[22] Neunert M, Abdolmaleki A, Wulfmeier M, Lampe T, Springenberg T, et al. Continuous-discrete reinforcement learning for hybrid control in robotics. In: Conference on robot learning. PMLR; 2020, p. 735–51.

[23] Li H, Wan Z, He H. Real-time residential demand response. IEEE Trans Smart Grid 2020;11(5):4144–54. http://dx.doi.org/10.1109/TSG.2020.2978061.

[24] 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. http://dx.doi.org/10.1109/TII.2022.3166215.

[25] Silver D, Lever G, Heess N, Degris T, Wierstra D, et al. Deterministic policy gradient algorithms. In: Xing EP, Jebara T, editors. Proceedings of the 31st international conference on machine learning. 32 of proceedings of machine learning research, Bejing, China: PMLR; 2014, p. 387–95.

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

[27] Williams RJ. Simple statistical gradient-following algorithms for connectionist reinforcement learning. Mach Learn 1992;8(3–4):229–56.

[28] Tensorflow Keras Optimizers: RMSprop. https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/RMSprop.

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

[30] Van Cutsem T, Glavic M, Rosehart W, Canizares C, Kanatas M, et al. Test systems for voltage stability studies. IEEE IEEE Trans Power Syst 2020;35(5):4078–87.

[31] 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. http://dx.doi.org/10.1016/j.epsr.2021.107581.

[32] 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. http://dx.doi.org/10.1016/j.ijepes.2006.03.005.