Deep Reinforcement Learning for Long-Term Voltage Stability Control

Deep Reinforcement Learning for Long-Term Voltage Stability Control

Hannes Hagmar
Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden
hannes.hagmar@chalmers.se

Le Anh Tuan
Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden
tuan.le@chalmers.se

Robert Eriksson
Department of Power Systems
Swedish National Grid
Sundbyberg, Sweden
robert.eriksson@svk.se

***Abstract*—Deep reinforcement learning (DRL) is a machine learning-based method suited for complex and high-dimensional control problems. In this study, a real-time control based on DRL is developed for long-term voltage stability control. The possibility of using future system services from demand response (DR) and energy storage systems (ESS) as control actions to stabilize the system is investigated. The performance of the DRL control is evaluated on a modified version of the Nordic32 test system. The results show that the DRL control quickly learns an effective control policy that can handle the uncertainty involved in using DR and ESS. The DRL control is compared to that of a rule-based load shedding scheme and the DRL control is shown to stabilize the system both significantly faster and with less load curtailment. Finally, the robustness and the generalization capability of the control are evaluated by testing the performance on load and disturbance scenarios that were not included in the training data.**

***Index Terms*—Deep reinforcement learning, emergency control, voltage stability, optimal control, real-time control**

I. Introduction

Long-term voltage instability has caused several major blackouts in the past and is a major aspect of power system security assessment [1]. In the case of a larger disturbance, the ability for system operators to act quickly with the correct control measures is imperative to avoid a fully developed voltage collapse. Voltage stability control typically includes actions such as generation redispatch or tripping, load shedding, or controlled system separation [2]. Choosing efficient and suitable control actions, which can both mitigate instability *and* minimize the impact on the end-consumers, can significantly improve the operational efficiency during adverse events.

While some actions are automatically triggered by local protection schemes, some are required to be manually initiated by the system operators. Those protection actions are usually based on fixed settings that are determined through off-line simulations of a few anticipated contingency scenarios and forecasted system conditions [3]. Long-term voltage stability (LTVS) events are often deceiving and the system may seem stable only to end up in an unstable state within a short time [4]. Thus, once instability has been detected, the remaining time to evaluate the system condition and to choose suitable control actions is limited and can be overwhelming for system operators. The current system situation may also significantly differ from any of the previously studied off-line contingency scenarios and there is a risk that the actions that are taken are not sufficient in restoring the system's stability.

Power system control is a problem of dynamic and sequential decision-making under uncertainty [3]. Traditional methods based on optimal control (e.g. model predictive control), have difficulties in handling large dynamic models of real power systems. To be able to compute the optimal control actions in a time frame required by system operators, significant simplifications of the system model are then generally required. However, deep reinforcement learning (DRL), has in recent years shown significant progress in solving high-dimensional and complex control problems. It is based on having a control agent learn an optimal policy through interactions with a real power system or its simulation model [5], [6], where the combination with deep learning models allows it to handle large and continuous state spaces. Previous implementations of DRL in emergency control include methods for dynamic breaking [3], optimal load shedding for short-term voltage stability [3], [7], [8], automatic voltage control [9]–[12], and oscillation damping [13].

In this paper, we develop a DRL method for fast, optimal, and adaptive control for LTVS events. The method can in real-time suggest optimized control actions to system operators to stabilize the system. The DRL agent continuously monitors the system state, and if the taken actions are not sufficient, additional corrective actions are proposed. The main contributions are the following:

- A methodology for a DRL-based control for LTVS. The developed DRL agent can in real-time suggest optimized control actions to system operators to mitigate voltage instability. The problem formulation and the reward scheme are designed to incentivize a control that quickly stabilizes the system to a minimal cost.
- An evaluation of using future system services from e.g. demand response (DR) and energy storage systems (ESS) as a more economic and flexible alternative to stabilize the system. The paper specifically examines the capability of the DRL control to account for the uncertainty involved in using such services (e.g. the price and the availability) as an alternative to conventional load shedding.
- An evaluation of the method's robustness and capability of handling scenarios that have not been included in the

training data of the algorithm. This is important since the number of states and the possible combinations with different disturbance scenarios are very large for real power systems.

The rest of the paper is organized as follows. In Section II, the theory in DRL is presented. In Section III, the proposed method is presented along with the steps for developing the training data and the training of the DRL algorithm. In Section IV, the results and discussion are presented. Concluding remarks are presented in Section V.

II. Deep Reinforcement Learning and PPO

In DRL, a control agent uses its control policy to interact with an environment (or a system) to give a trajectory of states, actions, and rewards. The received reward - also commonly referred to as the reinforcement signal - is used to determine whether the taken actions were effective. Through continuous interactions with the environment the agent is then trained to maximize the expected sum of future rewards over time [14].

In this study, we assume to have a stochastic policy $\pi(a|s)$ which models the conditional distribution for action $a \in \mathcal{A}$ given a state $s \in \mathcal{S}$. At each time step t, the agent observes the current state s_t and samples an action a_t from the policy. Once the action is taken in the environment, it responds with a new state $s' = s_{t+1}$ determined by a state transition dynamics $p(s'|s,a)$, and a reward R_t. For a parametrized policy $\pi_\theta(a|s)$, the goal of the agent is to learn the optimal parameters θ^* that maximizes a defined objective function $J(\theta)$. The objective function is commonly defined to be *expected* return, where the return, denoted G_t^γ, is the total discounted reward from a time step t and onward:

$$G_t^\gamma = \sum_{k=t}^{T} \gamma^{k-t} R_t \tag{1}$$

and 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: $Q^\pi(s,a) = \mathbb{E}\left[G_t^\gamma | s_t = s, a_t = a; \pi\right]$.

The Proximal policy optimization (PPO) algorithm, first presented in [15], is a policy gradient method that learns a parameterized policy that can select actions without requiring a value function [16]. In this paper, we use the ”clipped” version of the PPO algorithm, where the objective is defined as:

$$J^{clip}(\theta) = \hat{\mathbb{E}}_t \left[\min\left(r_t(\theta)\hat{A}_t, \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t\right)\right] \tag{2}$$

where r_t is a probability ratio given by:

$$r_t = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \tag{3}$$

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. $\hat{A}_t$ is an estimator of the advantage function at time step t, given by

$$\begin{aligned} \hat{A}_t &= \hat{Q}_\phi^\pi(a_t, s_t) - \hat{V}_\phi^\pi(s_t) = \\ &= \underbrace{R_t + \gamma\hat{V}_\phi^\pi(s_{t+1}) - \hat{V}_\phi^\pi(s_t)}_{\delta\text{-error}} \end{aligned} \tag{4}$$

where $\hat{V}_\phi^\pi$ and $\hat{Q}_\phi^\pi$ are estimates of the value function and the action-value function, respectively. The action-value can be written recursively as the sum of the immediate reward R_t after taking action a_t in state s_t and the estimated discounted value of the subsequent state $\gamma\hat{V}_\phi^\pi(s_{t+1})$. When this recursive expression is used to form the advantage, it is commonly referred to as the δ-error (or temporal-difference error).

By computing the gradient of the objective function (typically by using automatic differentiation software such as Tensorflow or PyTorch), one can adjust the current policy through stochastic gradient ascent (or by alternatives such as Adam [17]) so that the defined objective function is maximized. The clipped objective function $J^{clip}(\theta)$ ensures that one does not move *too* far away from the current policy, which allows one 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*, thus making disadvantageous actions less probable to be chosen by the policy in the future.

The value function used to compute the advantage function in (4) is generally unknown and has to be learned simultaneously as the policy. If the value function is learned in addition to the policy and the δ-error is used to approximate the advantage function, the algorithm is usually referred to have an Actor-Critic architecture. The policy π_θ is estimated by the *actor* while the value function $\hat{V}_\phi^\pi$ is estimated by the *critic*. The value function can be learned by forming and minimizing a new cost function, $L(\phi)$, based on the mean-squared error (or some other loss function) of the sampled and computed δ-errors. In DRL, the capability of high dimensional feature extraction and non-linear approximation that deep learning and neural networks (NNs) provides is utilized. 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 thus to find the optimal node weights for these networks.

III. DRL for Long-term Voltage Stability Control

ADD SECTION HERE ABOUT THE GENERAL CONTROL PROBLEM FOR LTVS. DESCRIBE THAT THE POLICY DEVELOPED BY THE DRL The method of DRL control for LTVS is based on off-line training on a large data set consisting of dynamical simulations for a range of

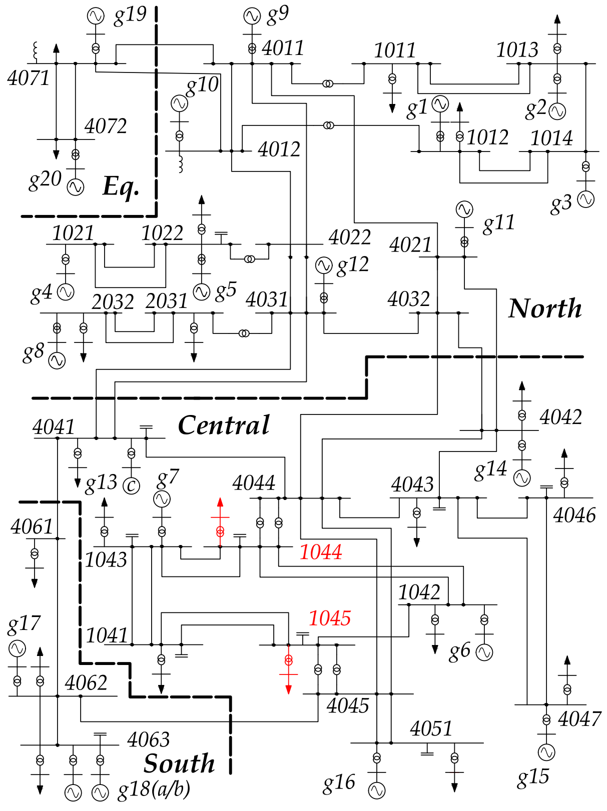

Fig. 1. One-line diagram of the modified Nordic32 system [19]. Load buses 1044 and 1045 that participate in the load curtailment are marked in red.

different disturbance and load scenarios. The training data are generated using PSS®E 35.0.0 with its in-built dynamical models [18]. All simulations have been tested on the slightly modified version of the Nordic32 test system, detailed in [19]. The test system is characterized by sensitivity towards long-term voltage instability. A one-line diagram of the test system is presented in Fig. 1. In the following sections, all the details in generating the training data and the development of the DRL control are presented.

A. Generation of training episodes

An overview of the steps involved in generating the disturbance scenarios and the training of the DRL agent is illustrated in Fig. 2. The different steps are detailed in the sections below.

1) Generation of initial operating conditions and disturbance scenarios: For the Nordic32 system, the initial operating conditions (OCs) were randomly generated around the insecure operation point denoted as "operating point A" in [19]. 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 (95 % of the original load as lower limit, 105 % of the original load as upper limit). The power factor of all loads was kept constant. A load flow solution was then computed where any changes in total load in the system were compensated by the slack bus generator, *g20*, see Fig. 1. Once an initial OC

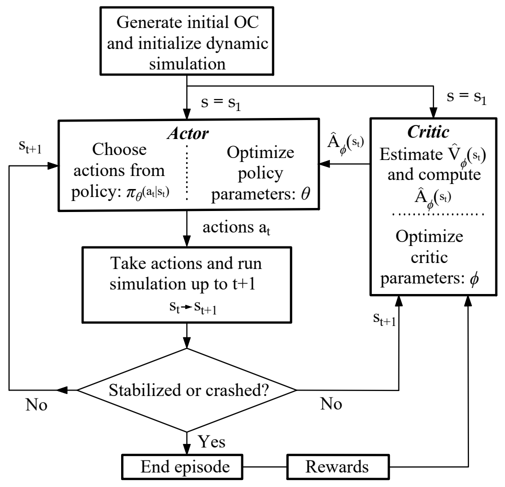
Generate initial OC and initialize dynamic simulation
s = s1
s = s1
Actor
Choose actions from policy: πθ(at|st)
Optimize policy parameters: θ
Critic
Estimate V̂φ(st) and compute Âφ(st)
Optimize critic parameters: φ
st+1
Âφ(st)
actions at
Take actions and run simulation up to t+1
st→st+1
st+1
Stabilized or crashed?
No
No
Yes
End episode
Rewards

Fig. 2. Flowchart showing the generation of training data and the interaction between the actor and critic network and the environment (the power system).

was generated, a dynamic simulation was initialized and a single larger disturbance was introduced. The DRL agent was trained to handle different types of disturbances, and with the same probability for each scenario, either a line was tripped between buses (i) *4032-4044*, (ii) *4032-4042*, (iii) *4031-4041*, (iv) *4021-4042*, or the tripping of either (v) generator *g6*, or (vi) generator *g7*. The disturbances were chosen as they were proven to cause significant system stress in the "Central" area (see Fig. 1), and without suitable control actions would in most load scenarios cause long-term voltage instability. In an actual application, preferably all disturbances that are likely to cause long-term voltage instability should be evaluated and included in the training. However, this would require significantly more training data to achieve satisfactory results and without a loss of generalization, we reduced the study to include only the previously mentioned disturbances.

2) Taking actions and continuing to the next time step: The state was then sampled from the system and passed to the actor network. The actor network outputs parameters that form the current policy $\pi_\theta(a|s)$ from which an action is sampled. That action was then activated in the system and the simulation continued to run until the next time step, which forms the state transition from $s_t \rightarrow s_{t+1}$. The time between each step in the simulation was 5 seconds (while the integration step size in the dynamical simulation in PSS®E was 0.05 seconds). The actions and states are further discussed in section III-C and III-B, respectively.

3) Evaluation of stability: Once the dynamical simulation reached the next time step, the stability of the system was evaluated. If any transmission system bus voltage (V_{TS}) were below 0.7 pu, the system was assumed to be unstable and the episode was terminated. All simulations ran for a maximum of 1000 seconds. If any V_{TS} at that time point was below 0.90 pu, the system was *also* assumed to be unstable.

4) Computing rewards and returns: Once the episode was finished, the rewards were computed. The reward R_t at each time step was a combination of the cost for the taken action

(C_a); a smaller penalty (-1) if any V_{TS} were below 0.90 pu; or a larger penalty (-500) if the system became unstable. The action cost C_a is further discussed in Section III-C. The reward at every time step was then computed as:

$$R_t = \begin{cases} C_a - (500 \cdot 0.99^t) & \text{if unstable} \\ C_a, & \text{else if all } V_{TS} \geq 0.90 \text{ pu} \\ C_a - (1 \cdot 0.99^t), & \text{else if any } V_{TS} < 0.90 \text{ pu} \end{cases} \quad (5)$$

The penalties were multiplied by a discounting factor of 0.99^t, resulting in lower negative penalties if instability and low system voltages occurred later rather than early in an episode. Once the reward for each step was computed, the returns G_t^γ were computed for all time steps according to (1). A $\gamma = 1$ was used since the actual cost of the taken actions will be the same no matter when they are taken during an episode. In this study, the reward is unitless, but should in real applications reflect the actual monetary cost of different actions and the corresponding rewards when the control goal is either achieved or missed.

B. States

The states were sampled from measurements taken from the dynamic simulation and consisted of a vector of i) bus voltages magnitudes, ii) active power flows, and iii) reactive power flows of all (and in between) buses in the system. While the relatively slow sample rate would allow measurements to be sampled from supervisory control and data acquisition (SCADA) systems, the availability and use of phasor measurement units would ensure a higher modeling accuracy through the time synchronized measurements. To also capture the dynamics of the system, *previous* observations from time step t-1 were stacked and included in the state vector (thus doubling the length of the state vector). To stabilize training, the state vectors were normalized by subtracting the mean value of each state value by its mean and then dividing by the 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 sampled states was stored. Once 10 000 sampled states were added to the list, the mean and standard deviation used for normalizing states became fixed.

C. Actions

To stabilize the system in case of instability, the DRL agent could activate load curtailment resources that, for instance, ESS and DR could provide to system operators. ESS and DR can essentially be viewed as available load curtailment that has been procured through a market system [20]. Using such services would allow system operators alleviate stress in a power system similar to that of load shedding. The difference is mainly i) that a higher degree of flexibility is available, where the activation level of the load curtailment based on ESS/DR can be typically be taken in much smaller steps than load shedding, and ii) that the impact on the end-users would be significantly less than when using forced load shedding.

The DR and ESS are modeled implicitly by allowing the DRL agent to adjust the load levels at two participating load buses within a certain range. The two load buses that participate are located at bus 1044 and bus 1045. The availability and the price of market-based system services provided by DR and ESS are typically varying; an uncertainty that needs to be included in the training of the DRL agent. To model this, the level of load curtailment that is available at each of the two participating buses is varied at the beginning of each disturbance scenario. The capacity of load curtailment is determined by sampling from a random uniform distribution with a lower level of 300 MW, and an upper level of 500 MW. Furthermore, the price of activating the load curtailment is also varied between the two participating buses which is achieved by randomly varying the price for each bus at the beginning of each disturbance scenario. The price of activating load curtailment at each bus is also determined by sampling from a random uniform distribution with a lower cost of -0.1/MW, and an upper level of -0.2/MW.

The DRL agent then controls the *total* level of load curtailment that is to be taken at each time step. The bus with the cheapest price is activated first, but if it has not sufficient capacity in adjusting its load, the other participating bus (with a higher price) will be activated as well. The range of the load curtailment capacity was chosen to ensure that all of the load and disturbance scenarios could be stabilized if sufficient load curtailment was utilized, while still adding an uncertainty in where the load curtailment was activated. The price variation models the market-based system, where the price of ESS/DR will typically vary depending on availability. Thus, the chosen approach ensures that there will be an uncertainty in *where* the actions are activated, to *what level* the actions are available at each bus, and also to *what cost* to the system. Depending on which bus the load curtailment is activated may also impact the effectiveness of the control action to mitigate instability, which is an additional uncertainty that the DRL agent need to account for.

D. Architecture and training of actor and critic network

The actor network, illustrated in Fig. 3 and further detailed in Table I, forms the mapping from states to the policy $\pi_\theta(a|s)$, from which actions are sampled. The network has two hidden layers followed by a final activation layer with two different activation functions used to form the outputs. The network outputs parameters used in defining a Normal distribution $\mathcal{N}$ from which the policy is defined and actions are sampled:

$$\pi_\theta(a|s) = \mathcal{N}\left(\mu_\theta(s), \sigma_\theta^2(s)\right) \quad (6)$$

The Normal distribution is parametrized by a mean value μ_θ and a standard deviation σ_θ, where the mean value μ_θ is computed using a linear activation function in the final layer, while the standard deviation σ_θ is computed using a softplus activation function that ensures that the value never becomes negative. The critic network is separate from the actor network and consists of a simple NN with a single hidden layer and a linear final activation function, further detailed in Table I.

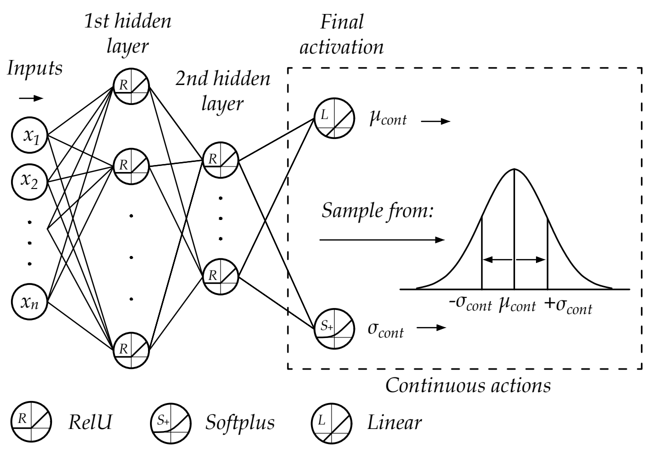

Fig. 3. Flowchart showing the generation of training data and the interaction between the actor and critic network and the environment (the power system).

TABLE I
DESIGN AND HYPERPARAMETERS USED IN TRAINING.

		Parameter	Values
Architecture	Critic	Number of inputs	976
		Neurons in hidden layer	128
		Final activation function	Linear
		Hidden layer activation	RelU
	Actor	Number of inputs	976
		Neurons in 1st hidden layer	64
		Neurons in 2nd hidden layer	32
		Final activation for μ_{cont}	Linear
		Final activation for σ_{cont}	Softplus
		Hidden layer activation	RelU
	Training	Max Epochs (K)	5
		PPO clip parameter (ϵ)	0.2
		Optimizer	Adam [17]
		Batch size (N)	64

Once a total of $N = 64$ episodes were sampled, the actor and the 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 cost function used to train the actor network is computed using (2) on all samples for all N episodes. Once the cost function for all samples was computed, the final $J^{clip}(\theta)$ was computed by taking the mean of those values. The cost function used to train the critic network, $L(\phi)$, is computed by taking the mean squared error on all δ-errors from (4) for all samples and all N episodes, followed by computing the mean of those values. The training was performed for $K = 10$ epochs on the whole batch of N episodes simultaneously. The values of the learning rates and other hyperparameters used in the training are specified in Table I.

IV. SIMULATIONS AND RESULTS

A. Training results

The DRL agent was trained for a total of 200 training iterations, corresponding to a total of 12 800 episodes, after which the performance converged. The training performance is shown in Fig. 4, where the episode rewards and whether the episode resulted in a crash or a stable state at the end of the simulation, is shown. The red line shows a centered moving average computed over the mean value over 250 episodes. The results in sub-figure (i) show that the performance improved rapidly until around 4 000 episodes, after which the policy managed to avoid system collapses completely. After this, the performance continued improving by mainly optimizing the level of action activation for each of the scenarios.

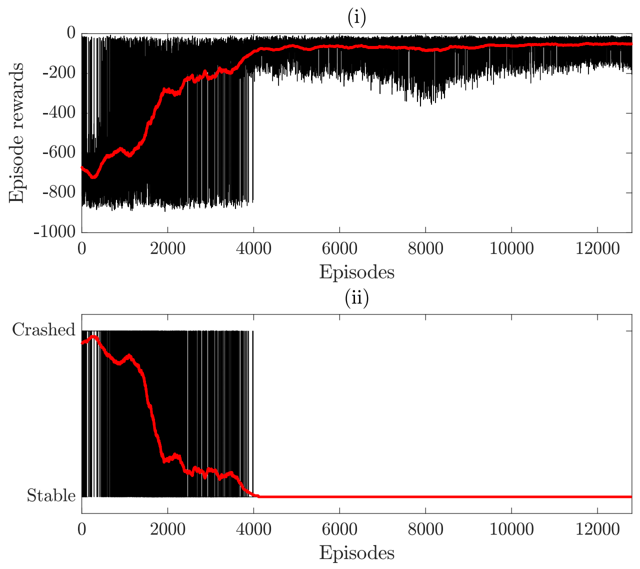

Fig. 4. Performance and development over episodes: Sub-figures showing (i) episode rewards during all episodes; (ii) whether the episode resulted in a crash or a stable state at the end of the simulation. The red line indicates a moving average computed over the mean of 250 data points.

B. Test results

During training, the DRL agent used a stochastic policy which allowed it to automatically explore the available action space, where exploration was governed by the parameter σ_θ. However, when using it online it is more suitable to transform the 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}. The trained DRL agent was tested on three different test sets. A total of 100 test scenarios were computed for each test set. The test sets used were defined as:

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 95 % to 105 % as specified in Section III-A, the OCs were adjusted randomly between 90 % to 110 %.
3) **Test set 3**: Introducing new unseen OCs by introducing a new disturbance that was not used in training the DRL agent. The new disturbance is the tripping of the line between the buses *4011-4021*. The same variation of

generation and load configuration as during training was used.

The performance of the developed DRL agent was also compared to that of a rule-based load shedding protection scheme. The load shedding protection scheme acts whenever *any* transmission system voltage is below 0.90 pu. In that case, a total of 100 MW load is removed from the system, divided equally between the loads located at bus 1044 and 1045. Once again, the power factor of each load is kept constant. To allow a fair comparison, the cost for activating the load shedding (C_a) was chosen to -0.15/MW, which is the mean value of the varying price for activation of the DR/ESS resources used by the DRL agent.

The average reward on the different test sets is presented in Table II and is computed as the mean episode reward of all test scenarios. In the final column, the relative difference between the DRL control and the load shedding scheme is presented. The results show that the DRL agent managed to get a significantly lower negative average reward compared to the load shedding control scheme on *all* different test sets. For instance, in test set 1, the load shedding scheme resulted in a 66.4 % higher *negative* reward compared to when the DRL control was. Although not being trained on the load and disturbance scenarios found in test set 2 and test set 3, the DRL agent managed to generalize its learning to these scenarios and still find a significantly more efficient control policy than for the load shedding scheme. The improvement was smallest on test set 3 (16.4 %) when a new disturbance that was not included in the training data was used to stress the system. It should be noted that all test scenarios were successfully controlled to stable states, both for the DRL control and the rule-based load shedding scheme.

In Table III, the average required load curtailment for each test set and control method is presented. This metric thus represents how much load each control method required to be curtailed before the system stabilized. Once again, the relative difference between the two control methods is presented in the final column in the table. The results show that the DRL agent required significantly less load curtailment to stabilize the system compared to the load shedding scheme, for all of the test sets. For instance, for test set 2, the load shedding required 178.7 % more load in average to be curtailed compared to what was used by the DRL control.

The differences between the DRL control and the control that is achieved with load shedding are exemplified in Fig. 5 and Fig. 6. In Fig. 5 the voltage magnitude at bus 1041 is shown for one of the test scenarios in test set 1. The voltage magnitude over time is presented for i) when the DRL control is used, ii) when the load shedding control is used, iii) and when no emergency control is used. In Fig. 6, the load at the controlled load buses 1044 and 1045 are also shown, which shows the difference in how the load is controlled by the DRL control and when using a load shedding control.

For the given scenario, the system will collapse after around 330 seconds if no control is initiated. For the case with load shedding, a total of 200 MW is shed from the system. The

TABLE II
AVERAGE PERFORMANCE REQUIRED FOR DIFFERENT TEST SETS AND CONTROL METHODS.

	Mean episode reward		Difference
	DRL control	*Load shedding*	[%]
Test set 1	-42.2	-70.2	66.4 %
Test set 2	-41.2	-82.9	101.2 %
Test set 3	-18.3	-21.3	16.4 %

TABLE III
AVERAGE LOAD CURTAILMENT REQUIRED FOR DIFFERENT TEST SETS AND CONTROL METHODS.

	Average load curtailment [MW]		Difference
	DRL control	*Load shedding*	[%]
Test set 1	190.0	452.0	137.9 %
Test set 2	192.0	535.1	178.7 %
Test set 3	123.2	137.5	11.6 %

load shedding is activated once at around 250 seconds, and then another activation occurs at around 380 seconds, which can be seen from the relatively large steps in load reduction in Fig. 6. After the second activation of the load shedding, the system voltages are restored in the system, which can be seen in Fig. 5. For the DRL control, the load curtailment is activated *directly* after the disturbance and in smaller increments, with no need to wait for the system to degrade before the control is activated. The load at bus 1045 is reduced by approximately 130 MW, after which the system is stabilized. The DRL control also manages to achieve a more satisfactory post-disturbance voltage magnitude profile, where the voltage magnitude is kept closer to the nominal pre-disturbance level. Thus, although the DRL control required a smaller amount of load curtailment, it achieved both a faster and more efficient control for the given scenario. The smoother control that is possible when utilizing load curtailment resources from DR and/or ESS also provided a more efficient way to mitigate voltage instability to a low system cost.

V. CONCLUSIONS

This paper introduces an optimal control method based on DRL to mitigate long-term voltage instability events in real-time. Once trained, the DRL control can continuously assess

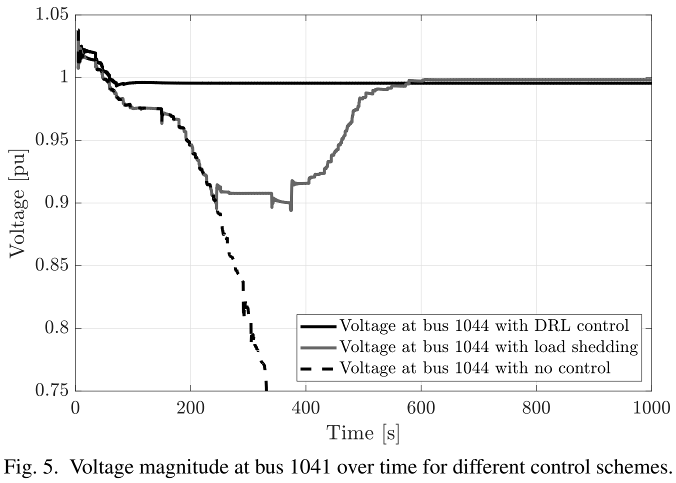

Fig. 5. Voltage magnitude at bus 1041 over time for different control schemes.

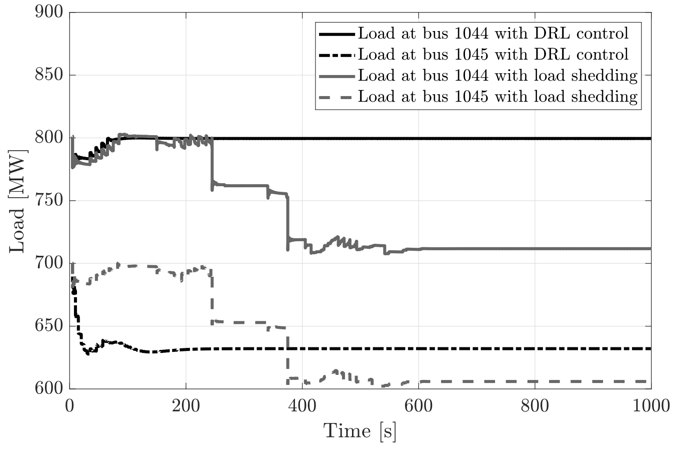

Fig. 6. Load development at bus 1044 and bus 1045 for the developed DRL control and for a load shedding control scheme.

the system stability and suggest fast and efficient control actions to system operators. The DRL control is trained to use system services from DR and ESS as a more efficient and flexible alternative to stabilize the system, compared to e.g. load shedding. The uncertainty in availability and the price of such market-based system services is modeled in the system. The developed DRL control was tested on a modified version of the Nordic32 test system and showed good performance on all of the developed test sets. The DRL control was also compared to a more conventional rule-based load shedding protection scheme and was shown to provide a more efficient and fast control.

Future research work includes: i) extending the study to include more actions spaces; ii) further evaluating the generalization capability of DRL control to handle scenarios not included in the training; iii) evaluating recent advancements in safe DRL to address control challenges in safety-critical systems such as power systems.

Acknowledgment

The work presented in this paper has been financially supported by Energiforsk and Svenska kraftnät (Swedish National Grid) under project numbers EVU10140 and EVU10450.

References

[1] T. Van Cutsem and C. Vournas, *Voltage stability of electric power systems*. Boston: Kluwer Academic Publishers, 1998.
[2] P. Kundur and G. Morison, "Techniques for emergency control of power systems and their implementation," *IFAC Proceedings Volumes*, vol. 30, no. 17, pp. 639–644, 1997.
[3] Q. Huang *et al.*, "Adaptive power system emergency control using deep reinforcement learning," *IEEE Trans. Smart Grid*, vol. 11, no. 2, pp. 1171–1182, 2020.
[4] M. Glavic and T. Van Cutsem, "A short survey of methods for voltage instability detection," in *Proc. (IEEE) PES General Meeting*, Detroit, MI, Jul 2011, pp. 1–8.
[5] D. Ernst, M. Glavic, and L. Wehenkel, "Power systems stability control: reinforcement learning framework," *IEEE Trans. Power Syst.*, vol. 19, no. 1, pp. 427–435, 2004.
[6] Z. Zhang, D. Zhang, and R. C. Qiu, "Deep reinforcement learning for power system applications: An overview," *CSEE Journal of Power and Energy Systems*, vol. 6, no. 1, pp. 213–225, 2020.
[7] J. Zhang *et al.*, "Deep reinforcement learning for short-term voltage control by dynamic load shedding in China southem power grid," in *2018 International Joint Conference on Neural Networks*, 2018.
[8] C. X. Jiang *et al.*, "Power system emergency control to improve short-term voltage stability using deep reinforcement learning algorithm," in *2019 IEEE 3rd International Electrical and Energy Conference (CIEEC)*, 2019, pp. 1872–1877.
[9] R. Diao *et al.*, "Autonomous voltage control for grid operation using deep reinforcement learning," in *2019 IEEE Power Energy Society General Meeting (PESGM)*, 2019, pp. 1–5.
[10] S. Wang *et al.*, "A data-driven multi-agent autonomous voltage control framework using deep reinforcement learning," *IEEE Trans. Power Syst.*, vol. 35, no. 6, pp. 4644–4654, 2020.
[11] B. L. Thayer and T. J. Overbye, "Deep reinforcement learning for electric transmission voltage control," in *2020 IEEE Electric Power and Energy Conference (EPEC)*, 2020, pp. 1–8.
[12] J.-F. Toubeau *et al.*, "Deep reinforcement learning-based voltage control to deal with model uncertainties in distribution networks," *Energies*, vol. 13, no. 15, p. 3928, 2020.
[13] Y. Hashmy *et al.*, "Wide-area measurement system-based low frequency oscillation damping control through reinforcement learning," *IEEE Trans. Smart Grid*, vol. 11, no. 6, pp. 5072–5083, 2020.
[14] D. Silver *et al.*, "Deterministic policy gradient algorithms," in *Proceedings of the 31st International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, E. P. Xing and T. Jebara, Eds., vol. 32, no. 1. Bejing, China: PMLR, 22–24 Jun 2014, pp. 387–395.
[15] J. Schulman *et al.*, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.
[16] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.
[17] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv e-prints*, p. arXiv:1412.6980, Dec 2014.
[18] *PSS®E 35.0.0 Model Library*, Siemens Power Technologies International, Schenectady, NY, Apr. 2019.
[19] T. Van Cutsem *et al.*, "Test systems for voltage stability studies," *IEEE Trans. Power Syst.*, vol. 35, no. 5, pp. 4078–4087, 2020.
[20] F. Rahimi and A. Ipakchi, "Demand response as a market resource under the smart grid paradigm," *IEEE Transactions on Smart Grid*, vol. 1, no. 1, pp. 82–88, 2010.