Model-Augmented Safe Reinforcement Learning for Volt-VAR Control in Power Distribution Networks

Yuanqi Gao[1], Nanpeng Yu[1]

Abstract

Volt-VAR control (VVC) is a critical tool to manage voltage profiles and reactive power flow in power distribution networks by setting voltage regulating and reactive power compensation device status. To facilitate the adoption of VVC, many physical model-based and data-driven algorithms have been proposed. However, most of the physical model-based methods rely on distribution network parameters, whereas the data-driven algorithms lack safety guarantees. In this paper, we propose a data-driven safe reinforcement learning (RL) algorithm for the VVC problem. We introduce three innovations to improve the learning efficiency and the safety. First, we train the RL agent using a learned environment model to improve the sample efficiency. Second, a safety layer is added to the policy neural network to enhance operational constraint satisfactions for both initial exploration phase and convergence phase. Finally, to improve the algorithm's performance when learning from limited data, we propose a novel mutual information regularization neural network for the safety layer. Simulation results on IEEE distribution test feeders show that the proposed algorithm improves constraint satisfactions compared to existing data-driven RL methods. With a modest amount of historical data, it is able to approximately maintain constraint satisfactions during the entire course of training. Asymptotically, it also yields similar level of performance of an ideal physical model-based benchmark. One possible limitation is that the proposed framework assumes a time-invariant distribution network topology and zero load transfer from other circuits. This is also an opportunity for future research.

Keywords: Volt-VAR control, data-driven, deep reinforcement learning, pathwise derivative, safe exploration

Nomenclature

Functions

$\hat{f}_\sigma(u_\psi)$	Approximated safety layer

[1]Department of Electrical and Computer Engineering, University of California, Riverside

$\mathbf{h}(x)$	Penalty function in the safety layer
$\phi(s)$	Voltage deviation estimation neural network
$\pi_\psi(s)$	Parameterized policy function
$f(u_\psi)$	Safety layer
$J(\phi)$	Loss function of the safety layer
$J(\psi)$	Objective function of policy network
$J(T)$	Objective function for mutual information estimation
$J_\beta(\phi)$	Regularized loss function of the safety layer
$P(s'\|s,a)$	State transition probability
$q^\pi(s,a)$	Action value function
$q_\varphi(s,a)$	Estimated action value function
$r(s,a)$	Reward function
$v^\pi(s)$	State value function
Parameters	
β	Regularization constant of the safety layer
ΔV	Maximum allowable voltage deviation
σ	Penalty coefficient in the safety layer
σ_ξ	Exploration noise standard deviation
C^l, C^s, C^v	Cost coefficient of line loss, device switching, and voltage deviation
K	Total number of VVC devices
M^{cap}	Capacitor size
$M^{\text{reg}}, M^{\text{tsf}}$	Regulator and OLTC step size
r^ℓ, x^ℓ	Line resistance and reactance of branch ℓ
Sets	
$\mathcal{A}$	Action space
$\mathcal{D}$	Replay buffer
$\mathcal{E}$	Set of edges of distribution networks
$\mathcal{S}$	State space
$\mathcal{T}$	Optimization horizon
Variables	
$\boldsymbol{p}_t$	Real power injections of all nodes at time t
$\boldsymbol{q}_t$	Reactive power injections of all nodes at time t
$\boldsymbol{x}_t^c$	All device status at time t before rounding operation
$\boldsymbol{x}_t$	All devices status at time t
ξ	Exploration noise
a	Action
A_t	Action at time t
l_t^{ij}	Squared current magnitude from node i to node j at time t
p_t^i	Real power injection of node i at time t
p_t^l	Total active power loss at time t
p_t^{ij}	Real power flow from node i to node j at time t
q_t^i	Reactive power injection of node i at time t
q^{i,cap_t}	Capacitor reactive power of node i at time t
q_t^{ij}	Reactive power flow from node i to node j at time t
R_t	Reward at time t

s	State
S_t	State at time t
u_ψ	Primitive action before the safety layer
V_t^i	Voltage magnitude of node i at time t
x_t^{cap}	Capacitor status at time t
x_t^{reg}	Tap position of voltage regulator at time t
x_t^{tsf}	Tap position of OLTC at time t

1. Introduction

Volt-VAR control (VVC) is one of the key technologies for voltage and reactive power management in smart grids. It determines the set points of voltage regulating devices such as voltage regulators and on-load tap changers (OLTCs), as well as reactive power compensation devices such as capacitor banks to control voltage profiles and reactive power levels. VVC is an indispensable tool for power distribution system management and is implemented in many distribution systems worldwide. Most electric utility companies adopt the legacy VVC approach, where the substation capacitors and field capacitors are controlled by a set of simple rules (for example, switch the capacitor on if the voltage is below a threshold and switch the capacitor off if the voltage is above another different threshold); the voltage regulator and OLTCs are controlled such that an estimated voltage level at the load center is maintained within a specific band. These control approaches have served the industry for many years and were effective in the past. However, the high penetration of distributed energy resources (DERs) and electric vehicles (EVs) brings about additional challenges such as rapid voltage fluctuations and reversed power flow [1]. These challenges make the aforementioned legacy control approaches insufficient.

More advanced control schemes and VVC algorithms have been studied in the literature. The concept of physical model-based VVC provides a number of benefits over the legacy ones. It computes the control actions by power flow based algorithms using the field measurements collected by the distribution management system (DMS) metering and communication facilities. This scheme allows closed loop, coordinated control based on the system-wide operating conditions. Recently, the physical model-based VVC algorithms have been further enhanced by leveraging mathematical programming and heuristic optimization techniques. A sensitivity analysis-based algorithm is proposed to improve the standard discrete coordinate-descent (DCD) VVC implementation [2]. The VVC has also been formulated as a deterministic optimization problem and solved using mixed-integer linear programming (MILP) [3], mixed-integer quadratically constrained programming (MIQCP) [4], and bi-level mixed-integer programming [5] algorithms. The deterministic formulation of VVC has been extended to robust optimization or stochastic programming formulations to handle uncertainties in loads/distributed generations (DGs) [6] [7] [8] [9]. Metaheuristic algorithms such as genetic algorithm [10], particle swarm optimization (PSO) [11], and parallelized PSO with high performance computing [12] have been proposed.

The coordinated control of slow timescale VVC devices and fast timescale inverters have been studied in two-timescale VVC frameworks. Stochastic programming with scenario reduction techniques have been proposed [13] [14] for two-timescale control. Local controls for grid edge VVC devices are also considered [15]. In addition to the two-timescale optimization, a third stage real-time droop controller is designed to mitigate the local voltage violation [16]. A modified alternating direction method of multipliers (ADMM) algorithm is leveraged to develop a bus-level distributed two-timescale smart inverter control strategy [17].

Physical model-based distributed algorithms for VVC have also been proposed to overcome the communication bottleneck. Simulated annealing-based methods [18] and distributed decision making algorithm [19] has been developed. The ADMM algorithm is leveraged for distributed discrete control by using continuous relaxation of discrete control variables [20]. The control variables can then be discretized by the adaptive threshold discretization technique [21].

Despite the theoretical advantages offered by physical model-based VVC algorithms, they have a number of practical limitations. First, physical model-based methods require that the distribution network/load models are available and accurate. However, the network data in the utility companies' geographic information system (GIS) are usually neither complete nor accurate [22]. It is a challenging task to build an accurate and reliable physical model of distribution network to apply the physical model-based VVC algorithms. Second, the computation time of many physical model-based algorithms remains a bottleneck due to the difficult underlying mixed-integer programs. This problem becomes more pronounced for large-scale distribution networks.

To this end, data-driven algorithms, which leverage advanced signal processing or artificial intelligence techniques, have been proposed. These methods work by learning VVC control strategies from online or historical operational data. Reference [23] combined a support vector regression (SVR) modeling the power flow equation with a model predictive control (MPC) framework. A k-nearest neighbor (kNN) power loss and voltage change estimator is combined with a heuristic approach to determine the device status [24]. When considering device switching cost, the VVC problem becomes a sequential decision making task with uncertainties. Many reinforcement learning (RL)-based VVC frameworks have been proposed in which the uncertainty is accounted by estimating the expectation of the objective. A consensus-based distributed tabular Q-learning algorithm is developed to solve the VVC problem [25]. The consensus protocol has been combined with maximum entropy RL to balance the exploration-exploitation [26]. Reference [27] developed a multi-agent deep Q-network (DQN) framework by decoupling the large combinatorial action space. To improve the RL performance when learning from limited data, a batch RL algorithm with data augmentation is proposed in [28]. Reference [29] achieved physical-model-free VVC by training a surrogate model for the distribution network and an RL algorithm utilizing the trained surrogate model. To improve the safety of the RL algorithms, [30] extend the traditional Markov decision

process (MDP) formulation of VVC as a constrained MDP. A safe off-policy RL algorithm is developed to improve the operational constraint satisfaction.

In contrast to physical model-based VVC methods, data-driven methods could achieve coordinated control without requiring accurate and complete system parameter information. Nonetheless, they still face practical implementation challenges related to safety and sample efficiency. Most of the data-driven methods do not have an explicit mechanism to ensure the safety of the action (e.g., whether a particular combination of tap positions will result in voltage dropping below the recommended range).

Some literature have considered the safety aspect of RL in VVC problems [30] and other power system applications. A backup controller is proposed for demand response [31] and service restoration [32]. The control Lyapunov functions is developed to pre-determine safe actions in damping power system oscillations [33]. Barrier functions are added to the reward for safe load shedding [34]. To improve the safety of resource dispatch, policy neural network is trained using the gradient of the Lagrangian relaxation function for real-time optimal power flow [35]. However, these methods either require an accurate system model or have weak constraint satisfaction performance during early stage of learning. Other than the safety issue, many data-driven methods needs to learn from a large amount of training data. Thus, it is difficult to apply data-driven methods when the amount of training data is small or covers a small region of the operating scenarios. In this case, the model could overfit the training data [36].

In this paper, we propose a model-based reinforcement learning (RL) algorithm with state-wise safety constraint for the Volt-VAR control problem. The main differences between our proposed RL method and the existing RL-based VVC studies are two-fold. First, we leverage model-based RL training. This method is more sample-efficient than the existing model-*free* RL training methods because model-based RL learns an environment model, then derives a policy from the model, whereas model-free RL directly learns policies from data. Note that by model-based we mean a supervised learning model (e.g. a neural network) that represents the VVC operating environment rather than the otherwise unavailable distribution system physical model itself. Second, we strategically incorporate a quadratic programming (QP)-based neural network layer in the RL model to execute actions that are safe for each state. This safety layer significantly improves the operating constraint satisfaction performance over the existing standard RL models. To improve the quality of the constraint function estimate, we propose an information-theoretic regularizer to overcome the overfitting issue for the constraint layer. This regularizer allows the constraint layer to learn accurate representation of the constraint function, while minimizing the overfitting caused by the correlated samples in the training dataset. The QP-based safety layer properly addresses the early safety issues of the existing asymptotically safe RL approach [30]. Simulation results on three IEEE test feeders show that the proposed algorithm yields similar asymptotic performance compared to other RL and physical model-based benchmarks. What's unique for the algorithm is it can maintain the voltage magnitude within the

acceptable range during the entire learning process.

The contributions of this paper are as follows:

• This paper proposes a model-augmented RL algorithm for the VVC problem to boost the sample efficiency.

• We propose a quadratic programming-based policy neural network architecture to enhance the safety.

• A mutual information regularizer is proposed to improve the quality of the constraint layer accuracy when learning from limited or correlated training samples.

The remainder of the paper is organized as follows: Section II presents the VVC problem formulation. Section III provides the technical methods. Section IV discusses the setup and results of experimental studies. Section V provides the conclusion.

2. Problem Formulation

In order to present the proposed model augmented reinforcement learning VVC framework, we first present the mathematical formulation of the VVC problem, then formulate the problem as a Markov decision process (MDP).

2.1. Mathematical Formulation of VVC

We consider a radial distribution network with nodes numbered from 1 to N. The 1-st node corresponds to the substation. The set of branches is denoted as $\mathcal{E}$. The nodal voltage magnitude, real and reactive power injection at time t is denoted as V_t^i, p_t^i, and q_t^i; the branch real and reactive power flow of line ij is denoted as p_t^{ij} and q_t^{ij}. The Volt-VAR control is done by controlling three types of devices listed as follows. First, a voltage regulator is placed at the substation node. The voltage regulator sets the discrete tap positions and produces a variable sets of reference voltages $V_t^1 = 1\text{p.u.} + x_t^{\text{reg}} \cdot M^{\text{reg}}$. x^{reg} is the tap position; M^{reg} is the fixed step size. Second, one or more on-load tap changers (OLTCs) are connecting certain feeder portions, which are modeled as a transformer with a variable turns ratio. The branch power flow for lines with OLTC is given by [8]:

$$(V_t^j/u_t)^2 = (V_t^i)^2 - 2(r^\ell p_t^{ij} + x^\ell q_t^{ij}) + [(r^\ell)^2 + (x^\ell)^2] l_t^{ij}, \tag{1}$$

where l_t^{ij} is the current magnitude squared; $u_t = 1 + x_t^{\text{tsf}} \cdot M^{\text{tsf}}$ is the turns ratio. Third, one or more capacitor banks are placed at certain nodes. The capacitor's reactive power is given by $q_t^{i,\text{cap}} = x_t^{\text{cap}} \cdot M^{\text{cap}} \cdot (V_t^i)^2$. The symbol $x_t^{\text{cap}} \in \{0, 1\}$ denotes the on-off status. Putting it all together, the power flow

model with VVC controls is given by (1)-(5).

$$p_t^i = \sum_{j:i\to j} p_t^{ij} - \sum_{j:j\to i} (p_t^{ji} - r^\ell l_t^{ij}) \quad \forall i = 2, ..., N \tag{2}$$

$$q_t^i + q_t^{i,\text{cap}} = \sum_{j:i\to j} q_t^{ij} - \sum_{j:j\to i} (q_t^{ji} - x^\ell l_t^{ij}) \; \forall i = 2, ..., N \tag{3}$$

$$l_t^{ij} = [(p_t^{ij})^2 + (q_t^{ij})^2]/(V_t^i)^2 \qquad \forall ij \in \mathcal{E} \tag{4}$$

$$V_t^1 = 1\text{p.u.} + x_t^{\text{reg}} \cdot M^{\text{reg}} \tag{5}$$

The total active power loss is calculated as $p_t^l = \sum_{i=1}^N p_t^i$. The VVC problem can be formulated mathematically as:

$$\begin{aligned} \min_{\boldsymbol{x}_t, t\in\mathcal{T}} \quad & \sum_{t\in\mathcal{T}} C^l p_t^l + C^s |\boldsymbol{x}_{t-1} - \boldsymbol{x}_t| \\ \text{s.t.} \quad & \underline{V} \le V_t^i \le \bar{V} \quad \forall t \in \mathcal{T}, i = 2, ..., N \\ & (1) - (5) \quad \forall t \in \mathcal{T} \end{aligned} \tag{6}$$

where $\mathcal{T}$ is the optimization time horizon; $\boldsymbol{x}_t$ is the vector of all device status at time t; C^l and C^s are cost coefficients of network loss and device switching. To approach this problem using RL, we need to formulate it as a Markov decision process (MDP) below.

2.2. Review of MDP

An MDP $(\mathcal{S}, \mathcal{A}, P, r, \gamma)$ consists of a state space $\mathcal{S}$, an action space $\mathcal{A}$, a state transition probability $P(s'|s, a)$, a reward function $r(s, a) : \mathcal{S}\times\mathcal{A} \mapsto \mathbb{R}$, and a discount factor $\gamma \in (0, 1)$. An "agent" can interact with the MDP by taking actions A_t at each time step, the environment then provides the reward $R_{t+1} = r(S_t, A_t)$ to the agent and transitions to some other states S_{t+1}. The goal of the agent is to find a policy $\pi(a|s)$, which maps states to probability distribution of actions, such that the expected discounted return $v^\pi(s) = \mathbb{E}_\pi \left[\sum_{t=0}^T \gamma^t R_{t+1} | S_0 = s\right]$ is maximized. The notation $\mathbb{E}_\pi[\cdot|S_0 = s]$ means starting from state s and following the policy π thereafter; the discount factor γ controls the contribution of far future reward to the optimizing objective; T is the operating horizon which may be infinite. $v^\pi(s)$ is called state value function. A related function commonly used in reinforcement learning is the action value function defined as $q^\pi(s, a) = \mathbb{E}_\pi \left[\sum_{t=0}^T \gamma^t R_{t+1} | S_0 = s, A_0 = a\right]$.

2.3. MDP Formulation of VVC

The MDP formulation of the VVC is as follows.

Action: the action is continuous and multi-dimensional. Each action dimension corresponds to one VVC device. We denote the action as $A_t = \boldsymbol{x}_t^c = [x^{c,1}, x^{c,2}, ..., x^{c,K}]$. K is the total number of VVC devices. Since the device tap positions are discrete, when applied to device control, each dimension of the actions will be rounded to their nearest tap position.

State: the state is given by $S_t = [\boldsymbol{p}_t, \boldsymbol{q}_t, \boldsymbol{x}^c_{t-1}, t]$. The bold vectors group variables from all nodes and devices. For example, $\boldsymbol{p}_t$ is the real power injection of all nodes. We also include the previous action as part of the state.

Reward: the reward function is given by

$$R_{t+1} = -C^l p^l_t - \sum_k C^s |[x^{c,k}_{t-1}| - [x^{c,k}_t|| - C^v C_{t+1}, \tag{7}$$

where the $[\cdot|$ notation denotes rounding to the nearest tap position. For example, if the k-th device is an OLTC, then $[x^{c,k}_t| = x^k_t$. In this study, the number of tap position change is considered as part of the objective function rather than a hard constraint. In distribution system VVC, one of the primary objectives is to ensure acceptable voltage for all customers under various operating conditions [22]. Therefore, the term C_{t+1} which discourages voltage deviation from the flat voltage profile (1.0 per unit) is added:

$$C_{t+1} = \sum_{i=1,\dots,N} |V^i_t - 1.0| \tag{8}$$

The constant C^l, C^s, C^v are cost coefficients of each of the reward components. Their detailed value will be given in Section IV.A.

In the next section, we present the technical details of the proposed model-based RL algorithm.

3. Technical Methods

For the technical methods, we start with an overview of the proposed framework. Then we describe the individual components. We first review basics of model-based reinforcement learning and the model-augmented actor-critic algorithm. We then introduce a constrained policy network to improve the safety, along with a mutual information regularization technique to improve the constrained actor network's accuracy. Finally, we provide the implementation details.

3.1. Overview

This subsection provides an overview of the proposed RL based VVC framework. This consists of the environment model, the actor-critic architecture, the constrained neural network layer, and the mutual information regularizer as shown in Fig. 1. The Supervisory Control and Data Acquisition (SCADA) data including real power and VVC device tap, as well as advanced metering infrastructure data including real power and voltage magnitude, are first collected by the DMS and converted to the appropriate format for algorithm training. Both historical and online data will be leveraged for this framework. In the model-based RL block, these operational data will be used to train the model $p_\theta(s'|s,a), r_\theta(s,a)$, whose output will be used to train the critic network $q_\varphi(s,a)$ and the actor (policy) network $\pi_\psi(u|s)$. We need to train an environment model

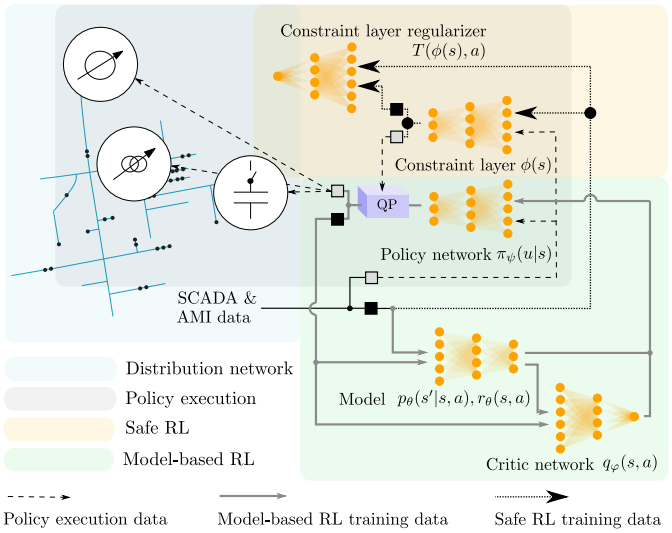

Fig. 1. Safe model-based RL for VVC

from operational data because in practice the system model is often unknown. This environment model is critical to improving the training efficiency for the RL algorithm. The output of the policy network u is a primitive action. The key to our proposed safe RL is a quadratic programming (QP) layer $f(u)$ which corrects the primitive action u such that the operational constraints are satisfied. The final corrected action is denoted by $a = f(u)$. In other words, the action a is the output of the RL-based VVC controller. We denote the composition of the policy network $\pi_\psi(u|s)$ and the QP $f(u)$ as $\pi_\psi^f(a|s) = f\left(\pi_\psi(u|s)\right)$. As part of the inputs to the QP layer, the voltage constraint function estimator network $\phi(s)$ and its corresponding regularizer network $T(\phi(s), a)$ will be trained using the operational data. The regularizer network improves the constraint network's performance when the training data is limited. In the next subsections, we will describe each component of this framework.

3.2. Overview of Model-Based Reinforcement Learning

Model-based RL trains a parametric model for the environment from which the value/policy functions are learned. Model-based RL is sample efficient for the VVC problem because it separates an RL problem into a model learning problem (a supervised learning task) and a policy learning problem. Additionally, model-based RL allows the environment prior information to be incorporated in the model learning phase. Section III.F will explain how the prior information about the VVC problem facilitates the model design. Nevertheless, model-based RL's performance is held back by the model accuracy. As such, we need to make the asymptotic performance the most important design criterion when developing and tailoring model-free RL methods for VVC problem.

In this paper, we denote the parametric model of the VVC environment as $p_\theta(s'|s,a)$ and $r_\theta(s,a)$. $\theta = [\theta_p, \theta_r]$ groups two sets of parameters, one for the transition probability p and one for the reward r. The exact form of the parameterization will be presented in Section III.F. To develop tractable policy learning algorithms, we also parameterize the value function and policy as $q_\varphi(s,a)$ and $\pi^f_\psi(a|s)$ by artificial neural networks. Two important equations, namely the n-step Bellman equations for the value functions are as follows:

$$v^\pi(s) = \mathbb{E}_\pi[\sum_{t=0}^{n-1} \gamma^t R_{t+1} + \gamma^n v^\pi(S_n)|S_0 = s] \tag{9}$$

$$q^\pi(s,a) = \mathbb{E}_\pi[\sum_{t=0}^{n-1} \gamma^t R_{t+1} + \gamma^n q^\pi(S_n, A_n)|S_0 = s, A_0 = a] \tag{10}$$

for $n = 1, 2, ...$ Note that when $n = 1$, (9)-(10) become the conventional one-step Bellman equations. They will be referred to by later sections.

3.3. Model-Augmented Actor-Critic

Model-augmented actor-critic (MAAC) [37] is a model-based RL algorithm that learns the policy rapidly by training the value and policy networks using the gradient information of the environment model. Specifically, we start from a state S_0 experienced in the environment, then take an action according to the policy $A_0^\psi \sim \pi^f_\psi(a|S_0)$ (we use superscript ψ for the action A_0^ψ to emphasize its dependency on ψ). Then we utilize the environment model parameterized by θ to generate a reward and a next state, denoted as $R_1^{\theta,\psi} = r_\theta(S_0, A_0^\psi)$ and $S_1^{\theta,\psi} \sim p_\theta(s'|S_0, A_0^\psi)$. The above process is repeated starting from $S_1^{\theta,\psi}$ and so forth. Thus we obtain the following parametric trajectory τ:

$$\tau = S_0, A_0^\psi, R_1^{\theta,\psi}, S_1^{\theta,\psi}, \cdots, A_{H-1}^{\theta,\psi}, R_H^{\theta,\psi}, S_H^{\theta,\psi}, A_H^{\theta,\psi}, \tag{11}$$

where H is the length of the trajectory which is a tunable hyperparameter. The value of H will be given in Section IV. Based on the n-step Bellman equation (10), the objective of the policy neural network is formulated as:

$$\max_\psi \hat{J}(\psi, \tau) = \max_\psi \sum_{t=0}^{H-1} \gamma^t R_{t+1}^{\theta,\psi} + \gamma^H q_\varphi(S_H^{\theta,\psi}, A_H^{\theta,\psi}) \tag{12}$$

$\hat{J}(\psi,\tau)$ is an estimation of the right hand side of (10). To reduce the variance of the gradient we collect multiple parametric trajectories and average them $J(\psi) = \hat{\mathbb{E}}_\tau \hat{J}(\psi,\tau)$ before taking the maximum (the symbol $\hat{\mathbb{E}}$ denotes the average). The derivative of $J(\psi)$, called pathwise derivative, will be used to update the actor (policy) network's parameter:

$$\psi \leftarrow \psi + \delta \nabla_\psi J(\psi), \tag{13}$$

where δ is the step size parameter. (13) is the *policy improvement* step. Similarly, the critic network's weights will also be learned based on the n-step Bellman equation:

$$\varphi \leftarrow \varphi - \delta \nabla_{\varphi} \hat{\mathbb{E}}_{\tau}(q_{\varphi}(S_0(\tau), A_0(\tau)) - \hat{J}(\psi, \tau))^2, \tag{14}$$

where $S_0(\tau)$ and $A_0(\tau)$ is the first state and action of the trajectory τ. (14) is the *policy evaluation* step. Thanks to the pathwise derivative, MAAC can provide actor neural network with strong learning signal hence significantly improve the learning efficiency. The only requirement is that parameterizations of both environment model and the actor-critic model are differentiable, and the derivatives are informative (i.e. non-zero). Thus, it is more convenient to work with continuous action space rather than discrete ones since sampling actions from a discrete distribution is a non-differentiable operation. When generating the training data for the neural network, the action is still continuous. It is when actuating the VVC device that the rounding operation takes place. Note that we can plug in a physical model for the reward and environment transition function to evaluate the gradient of (12). However, since the model is often unavailable to the electric utility companies, we instead train a parametric model. Nevertheless, we shall see in Section III.F that we can still optimize the design of the reward and environment transition model by utilizing the system information as much as possible.

When interacting with the environment, some exploration is needed to gather sufficient training data. To add stochasticity to the actor in the MAAC framework, one can use the maximum entropy regularization to the reward function to learn a stochastic policy [37]. Alternatively, we can learn a deterministic policy and add random noise $\xi \sim \mathcal{N}(0, \sigma_{\xi})$ after the policy neural network to make the action exploratory:

$$u_{\psi} = \pi_{\psi}(s) + \xi. \tag{15}$$

When combined with other algorithm components, the exploration scheme (15) is easier to implement and tune compared with the maximum entropy approach. Therefore we adopt the exploration scheme in this paper.

However, explorations can lead to unsafe actions particularly during early stage of learning. In the next subsection, we introduce a neural network mechanism to improve the safety of action exploration in (15).

3.4. Safe Exploration by Embedding Quadratic Programming Layers

In this subsection, we introduce a quadratic programming (QP) based neural network layer for the actor network to improve the safety of the exploration process. The QP layer is the key mechanism we use to achieve safe exploration and safe RL. It is added on top of the standard feedforward architecture of the policy neural network. Embedding optimization routines in neural networks have been described in some safe reinforcement learning literature [38, 39]. For

VVC problems, the safety refers to keeping all voltage magnitudes within the allowable range. That is:

$$|V_t^i(s,a) - 1| < \Delta V \ \forall i \ \Leftrightarrow \ \max_i |V_t^i(s,a) - 1| < \Delta V \tag{16}$$

We use $\Delta V = 0.05$ p.u. in this study. Other practical constraints such as power flow limit can be enforced in a similar way. In this paper, we propose the following method to enforce the voltage constraint. Let $u_\psi = \pi_\psi(s) + \xi$ be the output of the actor network, then u_ψ is further corrected by the final constraint layer:

$$a_\psi = f(u_\psi) \tag{17}$$

where f is given by the solution of the optimization problem:

$$\begin{aligned} f(u_\psi) = \underset{a}{\text{argmin}} \quad & ||a - u_\psi||^2 \\ \text{s.t.} \quad & \max_i |V^i(s,a) - 1| < \Delta V \\ & -1 \preceq a \preceq 1 \end{aligned} \tag{18}$$

The final layer $f(u_\psi)$ finds a perturbed action a that differs the least from the original network output u_ψ in terms of the 2-norm, such that the nodal voltages under a_ψ are within the operation limit. From the VVC perspective, a_ψ is the output of the VVC controller. If the function $\max_i |V^i(s,a) - 1|$ has been estimated before RL agent training, then the safety constraint can be significantly improved. While conceptually simple, this layer is difficult to implement for two reasons. First, since accurate and reliable distribution network parameters are difficult to obtain, constructing the voltage constraint function is intractable. Second, the problem has a quadratic cost function with multiple inequality constraints. There is no straightforward analytical solution, even with linearizations.

To resolve these difficulties, we propose to estimate the voltage constraint function using operational data, and employ iterative numerical algorithms to approximately solve (18). We create another neural network $\phi(s)$, which takes states as input and output $K+1$ values $\phi_0(s), \phi_1(s), ..., \phi_K(s)$, where K is the dimensionality of the action vector. Then we approximate the constraint function as:

$$\max_i |V^i(s,a) - 1| \approx \phi_0(s) + \sum_{k=1}^{K} \phi_k(s) a_k \tag{19}$$

That is, the constraint function is bilinear in $a = [a_1, a_2, ..., a_K]^\intercal$ and the nonlinear features $\phi(s)$ learned from the states s. In particular, $\phi_0(s)$ is the bias term for the regression model (19). We choose (19) to approximate $\max_i |V^i(s,a) - 1|$. This is because the function is linear in action and hence easier to formulate in optimization model. We can also write the LHS as $\phi(s)^\intercal [1; a]$. To obtain

these nonlinear features, the neural network $\phi(s)$ will be trained by supervised learning. The training data include power injections and VVC device tap positions, which forms s and a; as well as voltage magnitudes, which are used to calculate the regression target $\max_i |V^i - 1|$. All data are taken from SCADA and AMI. The training method will be introduced later. Then we replace the constraint function in (18) with $\phi(s)^\intercal[1; a] < \Delta V$ to obtain a standard quadratic programming (QP) problem.

Now to employ the iterative algorithm to solve the QP, the number of iterations should be kept small to save the computation during both forward pass and backpropagation. The popular interior point methods are also difficult to adopt in this context because finding a feasible initial solution creates additional computation burden. To this end, we apply the Newton's method to the penalty-approximated problem:

$$\begin{aligned}\hat{f}_\sigma(u_\psi) = \underset{a}{\operatorname{argmin}} \, ||a - u_\psi||^2 + \sigma \mathbf{h}(\phi(s)^\intercal[1; a] - \Delta V) \\ + \sigma \mathbf{h}(a - 1) + \sigma \mathbf{h}(-a - 1)\end{aligned} \tag{20}$$

The penalty function $\mathbf{h}$ should satisfy that $\mathbf{h}(x) > 0$ when $x > 0$ and that $\mathbf{h}(x) = 0$ when $x \leq 0$. It also needs to be continuously differentiable. In this paper we choose the squared maximum $\mathbf{h}(x) = \max(0, x)^2$. To solve (20), we start from the initial guess $a_\psi^{(1)} = u_\psi$, then fix a relatively small σ and perform certain number of Newton steps of the form:

$$a_\psi^{(\nu+1)} = a_\psi^{(\nu)} - \left[\nabla^2 \hat{f}_{\sigma_\nu}(a_\psi^{(\nu)})\right]^{-1} \nabla \hat{f}_{\sigma_\nu}(a_\psi^{(\nu)}) \tag{21}$$

where ν is the iteration count. After that we increase σ and continue the Newton process. The final iterate will be the output (the corrected action). Note that $a_\psi^{(\nu+1)}$ is differentiable with respect to $a_\psi^{(\nu)}$, thus the backpropagation applies. Fig. 2 illustrates this architecture. The sequence of values $\sigma_1, \sigma_2, ..., \sigma_k$ are hyperparameters and will be given in the experiment section.

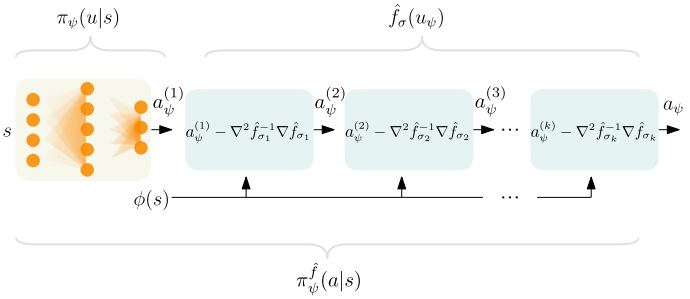

Fig. 2. Policy neural network architecture.

The penalty-based QP layer improves the safety of the actions produced by the actor network. However, a critical bottleneck of the constraint satisfaction

performance is the accuracy of the constraint function estimate (19). Standard training process for $\phi(s)$ would consists of the following cost function and the stochastic gradient descent:

$$J(\phi) = \hat{\mathbb{E}}_{(s,a,V^i)\sim\mathcal{B}}(\phi(s)^\intercal[1;a] - \max_i|V^i - 1|)^2 \tag{22}$$

$$\phi \leftarrow \phi - \delta\nabla_\phi J(\phi) \tag{23}$$

where $\mathcal{B}$ are mini-batch. We expect this estimate to be reasonably accurate if the training data are sufficient. Nonetheless, they can be limited in practice. The problem becomes even worse when the tap positions data lacks diversity or strongly correlates with the power injection patterns. In either case the neural network $\phi(s)$ can easily overfit.

3.5. Improving Constraint Layer Accuracy: The Minimum Mutual Information Regularization

To reduce overfitting, instead of using the standard mean-squared error loss function (22), we propose to regularize the neural network with the mutual information between the nonlinear feature and action. In order words, we train the $\phi(s)$ network by the following regularized loss function:

$$J_\beta(\phi) = J(\phi) + \beta I(\phi(s), a) \tag{24}$$

where the $J(\phi)$ term is defined in (22); the mutual information between two random vectors X and Y is defined as $I(X,Y) = \int_x \int_y p_{XY} \log \frac{P_{XY}}{P_X P_Y} dxdy$. Mutual information can be interpreted as the reduction of uncertainty of one random variable upon knowing another [40]. That is, it measures how much the two random vectors X, Y are dependent. The interpretation of the loss function (24) is as follows: we would like the nonlinear representation $\phi(s)$ and a to form an accurate prediction of the voltage magnitude constraint function (the first term); while the learned nonlinear representation $\phi(s)$ should not contain any irrelevant correlation with the tap position variables a (the second term). We suspect such correlation is one of the main reasons overfitting happens if trained by the first term only, which degrades the quality of the constraint satisfaction performance.

Mutual information regularization has been used as a reward regularization method in RL. Reference [41] studied the effect of adding mutual information between state and action to the reward function and its connection to the soft Bellman equation. However, mutual information is used in a different context in the present study. Here, we use it to regularize the training of the safety layer to reduce overfitting.

Since the computation of mutual information is intractable, to form a computable loss function we adopt the mutual information neural estimation (MINE) method detailed in [42]. We briefly introduce the MINE method here. The mutual information is first expressed as a dual representation:

$$I(\phi(s), a) \geq \sup_T \mathbb{E}_{p(\phi(s),a)}[T] - \mathbb{E}_{p(\phi(s))p(a)}[e^{T-1}] \tag{25}$$

where $T(\phi(s), a) \in \mathbb{R}$ is any admissible function. The first expectation is with respect to the joint distribution; the second one is with respect to the marginal distribution. Next, an estimator for $I(\phi(s), a)$ is constructed as follows. Let T be parameterized by a neural network, which is trained with the stochastic gradient ascent:

$$J(T) = \sum_{(s,a)\in\mathcal{B}} [T(\phi(s), a)] - \sum_{(s,G(a))\in\mathcal{B}} [e^{T(\phi(s),a)-1}] \tag{26}$$

$$T \leftarrow T + \delta \nabla_T J(T) \tag{27}$$

where $\mathcal{B}$ is the mini-batch. In the first term of (26), the state s along with a are jointly sampled from the training dataset, and then s is passed through the network ϕ; in the second term, a are randomly shuffled (denoted by the operator G) within the mini-batch to obtain samples from the marginal distribution. Upon convergence, the objective function $J(T)$ will serve as an estimate for the mutual information.

To combine the learning of the regularizer (26) with the objective function (24), we could iterate between the following two steps: first, the T network (26-27) is trained until converge; second, minimize (24) with the T network parameters fixed and use $J(T)$ as an estimate for $I(\phi(s), a)$. Alternatively, we could combine the two steps to train the two neural networks ϕ and T simultaneously as:

$$J_\beta(\phi, T) = J(\phi) - \beta J(T) \tag{28}$$

$$\phi \leftarrow \phi - \delta \nabla_\phi J_\beta(\phi, T(\phi)) \tag{29}$$

$$T \leftarrow T - \delta \nabla_T J_\beta(\phi, T) \tag{30}$$

We use the latter approach in this paper since it requires much less iterations and works equally well. The effectiveness of the regularization technique will be verified in Section IV.C.

3.6. Algorithm Implementations and Summary

In this section, we provide additional implementation details. Then we summarize the proposed model-augmented safe RL-based VVC algorithm.

3.6.1. Environment Model Architectures

To design the architecture for the environment model $p_\theta(S_{t+1}|S_t, A_t)$ we identify the estimation components. Recall that the state is defined as $S_t = [\boldsymbol{p}_t, \boldsymbol{q}_t, \boldsymbol{x}^c_{t-1}, t]$. As the transition for device status $\boldsymbol{x}^c_t$ and time index t are known and deterministic, we only need an estimator for the dynamics of $[\boldsymbol{p}_t, \boldsymbol{q}_t]$. As such we need consider the *aleatoric* uncertainty and the *epistemic* uncertainty [43]. The former describes the uncertainty inherent in the environment; the latter is the uncertainty caused by a lack of training data. To account for both uncertainties, we adopt the bootstrap ensemble of probabilistic neural networks [44] for the power injection time series model. That is, we train N neural networks $\theta = [\theta^1, \theta^2, \cdots, \theta^N]$ with identical architectures. Each neural network k

has a Gaussian output $[\boldsymbol{p}_{t+1}, \boldsymbol{q}_{t+1}] \sim \mathcal{N}(\boldsymbol{p}, \boldsymbol{q}; \mu_{\theta^k}(\boldsymbol{p}_t, \boldsymbol{q}_t, t), \Sigma_{\theta^k}(\boldsymbol{p}_t, \boldsymbol{q}_t, t))$, where Σ_{θ^k} is a diagonal matrix. The parameters θ is trained by the maximum log-likelihood:

$$P(\theta^k) = \mathcal{N}(\boldsymbol{p}_{t+1}, \boldsymbol{q}_{t+1}; \mu_{\theta^k}(\boldsymbol{p}_t, \boldsymbol{q}_t, t), \Sigma_{\theta^k}(\boldsymbol{p}_t, \boldsymbol{q}_t, t)) \tag{31}$$

$$\theta^k \leftarrow \theta^k + \delta \nabla_{\theta^k} \hat{\mathbb{E}} \log P(\theta^k) \tag{32}$$

To reduce the dimensionality of the input and output, the power injection pattern $[\boldsymbol{p}_{t+1}, \boldsymbol{q}_{t+1}]$ is projected to its first principal component before feeding into the neural network. The principal component approximation is legitimate if the power injections across different nodes are strongly correlated.

The state transition for the device status $\boldsymbol{x}^c_{t-1}$ and the time index t are known and deterministic. Thus we simply use their original arithmetic to calculate the transition, without any neural architecture. For device status we use $\boldsymbol{x}^c_t \leftarrow A_t$; for time index, we additionally use the cosine encoding $[\cos(\omega t), \sin(\omega t)]$ for t and update them according to the sum rule of trigonometry. ω is set to be the one week period $2\pi/168$. These arithmetics are continuously differentiable.

The structure of the reward function (7) allows us to optimize the model architecture as well. The switching cost term $\sum_k C^s |x^{c,k}_{t-1} - x^{c,k}_t|$ can be directly evaluated using simple arithmetics without an estimator. Therefore, we create a neural network to estimate the sum of costs due to loss and voltage deviation $C^l p^l_t + C^v C_{t+1}$; then we sum the output with the switching cost term as the overall model architecture. Note that we used the original continuous variable $x^{c,k}_t$ in the switching cost term without rounding. This is because we need a non-zero gradient when back-propagating through the model. Since the $C^l p^l_t + C^v C_{t+1}$ term is deterministic when given a state and action, we use an ensemble of deterministic neural networks $r_{\theta^k}(S_t, A_t), k = 1, 2, ..., N$ to only capture the epistemic uncertainty:

$$L(\theta^k) = (r_{\theta^k}(S_t, A_t) - C^l p^l_t - C^v C_{t+1})^2 \tag{33}$$

$$\theta^k \leftarrow \theta^k + \delta \nabla_{\theta^k} \hat{\mathbb{E}} \log L(\theta^k) \tag{34}$$

3.6.2. Safety Layer Architectures

The ensemble design can be carried over to the constraint function $\phi(s)$ as well. That is, the nonlinear feature network consists of an ensemble of feed-forward neural networks $\phi(s) = [\phi^1(s), \phi^2(s), ..., \phi^N(s)]$. As such, the voltage magnitude constraints becomes $\phi^k(s)^\intercal [1; a] < \Delta V$, $\forall k$. This leads to more conservative action choice compared to having only one constraint function. Accordingly, we create a separate regularizer network for each ϕ^k: $T(\phi(s), a) = [T^1(\phi^1(s), a), T^2(\phi^2(s), a), ..., T^N(\phi^N(s), a)]$. Each regularizer network is a standard feed-forward neural network.

3.6.3. Generating the Parametric Trajectories

The parametric trajectories (11) are generated as follows. We randomly sample a batch of states $\{s_i\}_{i=1,2,...B}$ from the replay buffer $\mathcal{D}$. For each s_i, we

take an action using the current policy $a_i = \pi_\psi^{\hat{f}}(s_i)$. Then we randomly sample a single model θ^k from the environment ensemble to evaluate the reward and next state: $r_i = r_{\theta^k}(s_i, a_i)$, $s_i' \sim p_{\theta^k}(s'|s_i, a_i)$. We continue this model rollout for H steps to obtain one parametric trajectory $\tau_i = s_i, a_i, r_i^1, r_i^2, ..., r_i^H, s_i^H, a_i^H$. Note that in each step, a new model is randomly sampled. The above process is carried out for each state in the minibatch to obtain the batch of trajectories $\{\tau_i\}_{i=1,2,...,B}$.

3.6.4. Algorithm Summary

The proposed model-based safety layer augmented RL algorithm for Volt-VAR control is summarized in Algorithm 1. The algorithm is first pre-trained on the historical dataset, then it will start controlling the VVC devices and continuing the learning indefinitely.

Algorithm 1 Safety layer augmented actor critic (SAAC) for VVC

Input: Historical dataset $\mathcal{D}$
Output: Policy $\pi_\psi^f(a|s)$

Initialize $p_\theta(s'|s,a)$, $r_\theta(s,a)$, $q_\varphi(s,a)$, $\pi_\psi(u|s)$, $\phi(s)$, $T(\phi(s),a)$
for $i = 1, ..., L$ **do** % pre-train
  Sample $\mathcal{B}_k = \{(s,a,r,s')\}_k \sim \mathcal{D}$, $\forall k = 1, ..., N$
  Train p_{θ^k} and r_{θ^k} on $\mathcal{B}_k$ by (32) and (34)
  Train ϕ^k and T^k on $\mathcal{B}_k$ by (29) and (30)
for $t = 1, ...,$ **do** % agent-environment interaction
  $S_t = [\boldsymbol{p}_t, \boldsymbol{q}_t, \boldsymbol{x}_{t-1}^c, t]$
  $A_t \sim \pi_\psi^{\hat{f}}(a|S_t)$
  $S_{t+1} = [\boldsymbol{p}_{t+1}, \boldsymbol{q}_{t+1}, A_t, t+1]$
  Obtain R_{t+1} by (7)
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{(S_t, A_t, R_{t+1}, S_{t+1})\}$
  for $i = 1, ..., l$ **do** % agent training
    Sample $\mathcal{B} = \{(s)\} \sim \mathcal{D}$
    Train π_ψ, q_φ by (13) and (14)
    Sample $\mathcal{B}_k = \{(s,a,r,s')\}_k \sim \mathcal{D}$, $\forall k = 1, ..., N$
    Train p_{θ^k} and r_{θ^k} on $\mathcal{B}_k$ by (32) and (34)
    Train ϕ^k and T^k on $\mathcal{B}_k$ by (29) and (30)

4. Numerical Studies

4.1. Distribution Test Feeders and Load Data

We use the IEEE 4-bus, 34-bus, and 123-bus radial distribution test feeders [45] for our numerical studies. For illustration purposes the test feeders have been converted to their positive sequence representations. However, the proposed algorithm is applicable for unbalanced systems as well. The RL-based VVC controller output is the action, which actuates the following VVC devices

of the feeders. The substation voltage (reference voltage) is controlled by an 11-step voltage regulator. The step size is $M^{\text{reg}} = 0.01$, which sets the reference voltage between 0.95 and 1.05 p.u. The OLTCs also have 11 tap positions with turns ratios ranging from 0.95 to 1.05. These OLTCs are placed at branch $(2, 3)$ for the 4-bus feeder, branches $(814, 850)$ and $(852, 832)$ for the 34-bus feeder, and branches $(10, 15)$, $(67, 160)$, and $(25, 26)$ for the 123-bus feeder. The capacitors have two tap positions corresponding to the on/off status. For the 4-bus feeder, a capacitor with rating $M^{\text{cap}} = 200$ kVar is placed at node 4; for the 34-bus feeder, two capacitors with rating 100 kVar and 150 kVar are placed at node 844 and node 847, respectively; for the 123-bus feeder, four capacitors are placed at node 83 (200 kVar), 88 (50 kVar), 90 (50 kVar), and 92 (50 kVar), respectively. Although all devices have discrete tap positions, they are internally represented as continuous variables by the RL algorithm. They will be rounded to the nearest tap positions when the action actuate the device. The energy cost C^l, switching cost C^s, and the voltage violation cost C^v are set as 40.0 \$/MWh, 0.1 \$/switching, and 1.0 \$/p.u., respectively. In practice, these values can be chosen according to the application requirements.

The London smart meter dataset [46] is used to represent the nodal power injection patterns. We convert the half-hourly kWh measurements to hourly values and take the first 4000 hours for our numerical study. The kWh measurements are then aggregated across the individual customers, and scaled appropriately to produce a realistic loading level for each test feeder. The spatial load distribution and power factors are set to be the same as that of the standard IEEE test feeders.

4.2. Algorithm Setup

The hyperparameter setup of the proposed algorithm and two benchmarks are provided in this subsection. The first benchmark is the discrete action space soft actor critic (SAC) [47] with action space decoupling and ordinal variable encoding [30]. The second benchmark is the vanilla model-augmented actor critic (MAAC) described in Section III.C, which is a simplified version of the proposed SAAC by removing the safety layer introduced in Section III.D. The hyperparameters are given in Table 1. The first section of Table 1 shows the parameters shared by all algorithms. We will use these parameters for all numerical studies. If a hyperparameter is chosen differenly based on the feeder, that parameter will be written in a curly bracket to represent the values for 4-bus, 34-bus, and 123-bus, respectively.

In this study, we assume that one thousand hours of operational data are available from the historical dataset before the agent start interacting with the system. The historical dataset collects the load/DG, voltage and network loss, as well as the corresponding tap positions. The purpose of this historical dataset is to mimic the database of a typical electric utility company. As such, we can pre-train certain components of the RL-VVC framework to provide a slightly better initial policy [48]. For the SAAC algorithm, we pre-train the environment model and the constraint layer using these historical data. To make the comparison

fair, the same set of historical data will be used to pre-train the policy and value networks for the benchmark algorithms SAC and MAAC as well.

Unfortunately, we were unable to find an open-source dataset for the device tap positions. Thus, we create them by following a mixed-integer conic programming (MICP) VVC algorithm. The created tap positions rarely change and correlate with the power injection pattern. As a result, the training samples have a significant lack of diversity and pose challenges for the RL algorithms. However, this is necessary to reproduce the difficulty of real world datasets.

The proposed SAAC algorithm and the SAC, MAAC baseline algorithms are implemented in Python using the PyTorch deep learning library [49]. The power flow and VVC program are also implemented in Python using the NumPy numeric computation library. The MICP and the model predictive control (MPC) benchmarks are implemented in MATLAB using the YALMIP optimization modeling toolbox [50].

4.3. Sample Efficiency and Constraint Satisfaction

The purpose of this subsection is to show the sample efficiency and constraint satisfaction performance of the proposed algorithm. As an example, Fig. 3-4 shows the voltage profile of the first and 16-th operating week for the 4-bus and 34-bus feeder. Each bar corresponds to 168 hours. The acceptable voltage range [0.95, 1.05] is highlighted by the dashed dark lines. Although all algorithms produce acceptable voltage profiles after 16 weeks of training, for the first operating week, the SAC and MAAC baseline algorithms need to explore actions with no guarantee of safe voltage level. On the other hand, the proposed SAAC maintains acceptable voltage levels during both weeks.

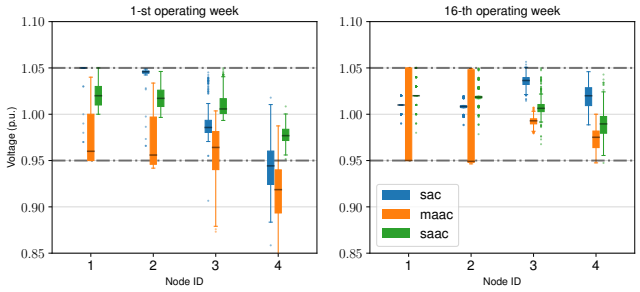

Fig. 3. Voltage profile of 4-bus system

More comprehensively, Fig. 5-7 plot the single time step reward and the maximum voltage magnitude deviation over the distribution network for all three test feeders. The maximum voltage magnitude deviation plot is added to help us assess the flatness of system-wide voltage magnitude, which is one of the most important criteria to evaluate the VVC performance. The solid curve is the average of five independent runs; the shaded regions are the corresponding error bounds. The reward curves in Fig. 5-7 show that the model-based reinforcement learning generally converge faster than the model-free SAC, even

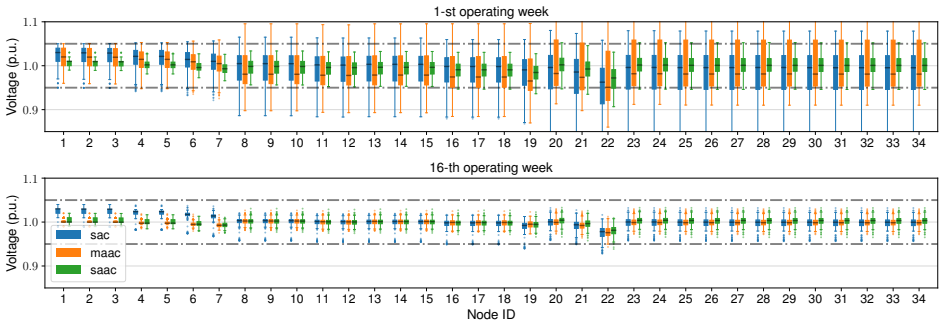

Fig. 4. Voltage profile of 34-bus system

if the latter has been pre-trained on the historical data. Asymptotically, the performance of model-based RL match the model-free SAC as well. Most importantly, the proposed SAAC has the best constraint satisfaction performance during the entire course of learning in all three test feeders. As indicated by the shaded regions, the constraint satisfaction performance is relatively consistent across different runs, although small violations can still happen occasionally. In contrast, other algorithms need to explore different actions and violate a certain number of constraints before converging to a useful policy.

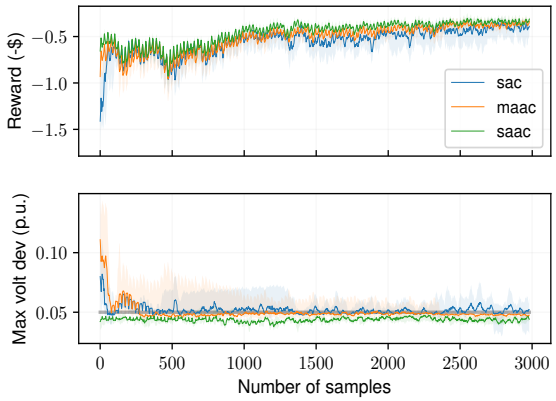

Fig. 5. Hourly reward and voltage deviation of 4-bus feeder.

To quantify the constraint satisfaction performance of SAAC, we provide a detailed look of how the QP-based constraint layer and the regularizer network affect the voltage magnitude deviation during early stage of learning. We first show the maximum voltage deviation estimate of (19) with and without the regularizer on a testing set. In order to mimic the exploratory behavior of RL agents, the tap position data of this testing set includes 70% randomly sampled taps with uniform distribution from the range of all taps, and 30% sampled using the same distribution as the training data. The results in Fig. 8 shows

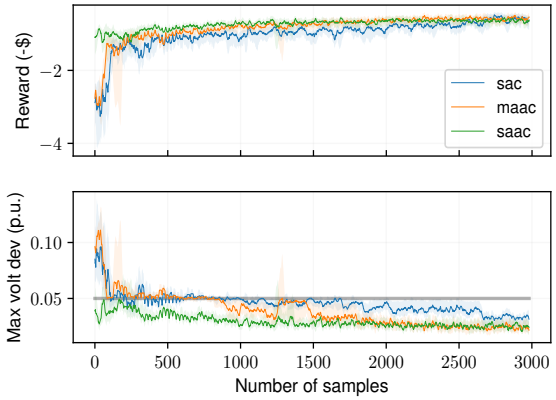

Fig. 6. Hourly reward and voltage deviation of 34-bus feeder.

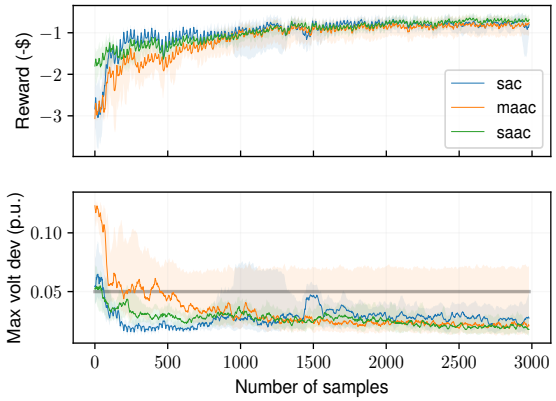

Fig. 7. Hourly reward and voltage deviation of 123-bus feeder.

that when combined with the mutual information regularizer, the out-of-sample testing performance has been improved significantly for all test feeders.

Next, we show the constraint satisfaction performance combined with the reinforcement learning agent. Starting with the policy network without any safety layer, we add the $\phi(s)$ based QP layer and the regularizer network $T(\phi(s), a)$ one by one and compare their voltage constraint satisfaction performance. Fig. 9 shows the maximum voltage deviation of the 34-bus system for the first 480 hours. The constraint satisfaction performance is improved appreciably by the QP-based constraint layer. With the regularizer network $T(\phi(s), a)$, the amount of voltage violation is further reduced, especially during early stage of learning. Although the two architectures cannot guarantee strict constraint satisfaction, the amount of violation is very small and acceptable in practice.

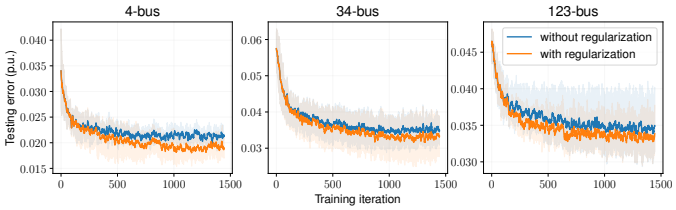

Fig. 8. Constraint function estimation with and without the regularization network.

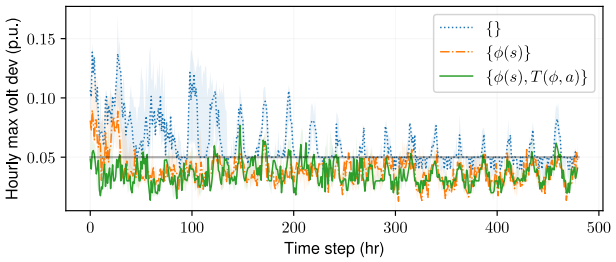

Fig. 9. Voltage deviations of 34-bus feeder for different policy networks: {} denotes no safety layer; $\{\phi(s)\}$ denotes with safety layer but no regularization network; $\{\phi(s), T(\phi, a)\}$ denotes with both safety layer and regularization network.

4.4. Comparison with Model-Based Benchmark

In this subsection, we evaluate the asymptotic performance of the proposed algorithm against the physical model-based benchmark, which approximate the ground truth solution of the VVC problem. Using the DistFlow equation, the VVC problem with objective function (7) can be formulated as a mixed-integer conic programming (MICP) model [8] [26], with the term $|V_t^i - 1|$ being replaced by $\frac{1}{2}|(V_t^i)^2 - 1|$. This approximation is very accurate for $0.95 \leq V_t^i \leq 1.05$. The MICP approach can be extended to a rolling horizon model predictive control (MPC) [51] framework to better capture the long term effect of the actions. In this work, we have used a horizon length of 10 for the 4-bus feeder and 2 for the 34-bus and 123-bus feeders. These values have been tested to produce results closer to the optimal than the MICP. We omit results for longer horizons since the increase of cumulative rewards is rather incremental while the computation time is significantly longer. We assume the ground truth load and distributed generations are known for both MICP and MPC models. We evaluate the negative of cumulative reward, cumulative voltage violation, and the total number of switches over the last 168 operating hours. The last 168 hour out of the 4000 hours are selected since the data-driven RL algorithms have converged to a relatively stable policy, thus are indicative for asymptotic performance. Table 2 shows that our proposed method yields similar level of asymptotic performance as that of the data-driven benchmark SAC and the physical model-based approaches.

5. Conclusion

This paper proposes a model-augmented safe reinforcement learning framework for the power distribution network Volt-VAR control problem. The framework addresses two of the most crucial problems with data-driven control methodologies: sample efficiency and safety. A model-augmented pathwise derivative is utilized to train the reinforcement learning algorithm to improve the sample efficiency. Then we embed an iterative quadratic programming based constraint satisfaction layer in the actor neural network to improve the safety. Finally, a novel mutual information regularizer is proposed to improve the performance of the constrained satisfaction layer. Simulation results confirm the superior sample efficiency and constraint satisfaction of our approach compared with other reinforcement learning benchmarks. In the future, we plan to improve the efficiency of the presented random exploration. We also plan to simplify the algorithm design and the associated parameter tuning process. One possible limitation is that the proposed RL-based VVC framework assumes a time-invariant distribution network topology and there is no load transfer between other circuits and the circuit being controlled. We will address this challenge in our future work.

References

[1] H. Sun, Q. Guo, J. Qi, V. Ajjarapu, R. Bravo, J. Chow, Z. Li, R. Moghe, E. Nasr-Azadani, U. Tamrakar, G. N. Taranto, R. Tonkoski, G. Valverde, Q. Wu, G. Yang, Review of challenges and research opportunities for voltage control in smart grids, IEEE Transactions on Power Systems 34 (4) (2019) 2790–2801. doi:10.1109/TPWRS.2019.2897948.

[2] R. A. Jabr, I. Džafić, Sensitivity-based discrete coordinate-descent for Volt/VAr control in distribution networks, IEEE Transactions on Power Systems 31 (6) (2016) 4670–4678. doi:10.1109/TPWRS.2015.2512103.

[3] A. Borghetti, Using mixed integer programming for the Volt/VAR optimization in distribution feeders, Electric Power Systems Research 98 (2013) 39–50.

[4] H. Ahmadi, J. R. Martí, H. W. Dommel, A framework for Volt-VAR optimization in distribution systems, IEEE Transactions on Smart Grid 6 (3) (2015) 1473–1483. doi:10.1109/TSG.2014.2374613.

[5] R. R. Jha, A. Dubey, C.-C. Liu, K. P. Schneider, Bi-level Volt-VAR optimization to coordinate smart inverters with voltage control devices, IEEE Transactions on Power Systems 34 (3) (2019) 1801–1813.

[6] N. Daratha, B. Das, J. Sharma, Robust voltage regulation in unbalanced radial distribution system under uncertainty of distributed generation and loads, International Journal of Electrical Power & Energy Systems 73 (2015) 516–527.

[7] W. Zheng, W. Wu, B. Zhang, Y. Wang, Robust reactive power optimisation and voltage control method for active distribution networks via dual time-scale coordination, IET Generation, Transmission & Distribution 11 (6) (2017) 1461–1471.

[8] F. U. Nazir, B. C. Pal, R. A. Jabr, A two-stage chance constrained Volt/VAR control scheme for active distribution networks with nodal power uncertainties, IEEE Transactions on Power Systems 34 (1) (2019) 314–325.

[9] Z. Zhang, F. F. da Silva, Y. Guo, C. L. Bak, Z. Chen, Double-layer stochastic model predictive voltage control in active distribution networks with high penetration of renewables, Applied Energy 302 (2021) 117530. doi:https://doi.org/10.1016/j.apenergy.2021.117530.
URL https://www.sciencedirect.com/science/article/pii/S0306261921009090

[10] T. Senjyu, Y. Miyazato, A. Yona, N. Urasaki, T. Funabashi, Optimal distribution voltage control and coordination with distributed generation, IEEE Transactions on Power Delivery 23 (2) (2008) 1236–1242.

[11] H. Yoshida, K. Kawata, Y. Fukuyama, S. Takayama, Y. Nakanishi, A particle swarm optimization for reactive power and voltage control considering voltage security assessment, IEEE Transactions on Power Systems 15 (4) (2000) 1232–1239.

[12] A. Anwar, A. N. Mahmood, J. Taheri, Z. Tari, A. Y. Zomaya, HPC-based intelligent Volt/VAr control of unbalanced distribution smart grid in the presence of noise, IEEE Transactions on Smart Grid 8 (3) (2017) 1446–1459. doi:10.1109/TSG.2017.2662229.

[13] Y. Xu, Z. Y. Dong, R. Zhang, D. J. Hill, Multi-timescale coordinated Voltage/Var control of high renewable-penetrated distribution systems, IEEE Transactions on Power Systems 32 (6) (2017) 4398–4408. doi:10.1109/TPWRS.2017.2669343.

[14] D. Jin, H. Chiang, P. Li, Two-timescale multi-objective coordinated Volt/Var optimization for active distribution networks, IEEE Transactions on Power Systems 34 (6) (2019) 4418–4428. doi:10.1109/TPWRS.2019.2914923.

[15] A. R. Malekpour, A. M. Annaswamy, J. Shah, Hierarchical hybrid architecture for Volt/Var control of power distribution grids, IEEE Transactions on Power Systems 35 (2) (2020) 854–863. doi:10.1109/TPWRS.2019.2941969.

[16] C. Zhang, Y. Xu, Z. Dong, J. Ravishankar, Three-stage robust inverter-based Voltage/Var control for distribution networks with high-level pv, IEEE Transactions on Smart Grid 10 (1) (2019) 782–793. doi:10.1109/TSG.2017.2752234.

[17] Q. Zhang, K. Dehghanpour, Z. Wang, Distributed CVR in unbalanced distribution systems with PV penetration, IEEE Transactions on Smart Grid 10 (5) (2019) 5308–5319. doi:10.1109/TSG.2018.2880419.

[18] A. Vaccaro, A. F. Zobaa, Voltage regulation in active networks by distributed and cooperative meta-heuristic optimizers, Electric power systems research 99 (2013) 9–17.

[19] H. E. Z. Farag, E. F. El-Saadany, A novel cooperative protocol for distributed voltage control in active distribution systems, IEEE Transactions on Power Systems 28 (2) (2013) 1645–1656.

[20] B. A. Robbins, H. Zhu, A. D. Domínguez-García, Optimal tap setting of voltage regulation transformers in unbalanced distribution systems, IEEE Transactions on Power Systems 31 (1) (2016) 256–267.

[21] F. U. Nazir, B. C. Pal, R. A. Jabr, Distributed solution of stochastic Volt/VAr control in radial networks, IEEE Transactions on Smart Grid 11 (6) (2020) 5314–5324. doi:10.1109/TSG.2020.3002100.

[22] Electric Power Research Institute, Design and assessment of Volt-VAR optimization systems, Technical Update (2011).

[23] E. Pourjafari, M. Reformat, A support vector regression based model predictive control for Volt-VAR optimization of distribution systems, IEEE Access 7 (2019) 93352–93363. doi:10.1109/ACCESS.2019.2928173.

[24] P. Bagheri, W. Xu, Model-free Volt-VAR control based on measurement data analytics, IEEE Transactions on Power Systems 34 (2) (2019) 1471–1482. doi:10.1109/TPWRS.2018.2874543.

[25] Y. Xu, W. Zhang, W. Liu, F. Ferrese, Multiagent-based reinforcement learning for optimal reactive power dispatch, IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 42 (6) (2012) 1742–1751. doi:10.1109/TSMCC.2012.2218596.

[26] Y. Gao, W. Wang, N. Yu, Consensus multi-agent reinforcement learning for Volt-VAR control in power distribution networks, IEEE Transactions on Smart Grid (2021). doi:10.1109/TSG.2021.3058996.

[27] Y. Zhang, X. Wang, J. Wang, Y. Zhang, Deep reinforcement learning based Volt-VAR optimization in smart distribution systems, IEEE Transactions on Smart Grid 12 (1) (2021) 361–371. doi:10.1109/TSG.2020.3010130.

[28] H. Xu, A. D. Domínguez-García, P. W. Sauer, Optimal tap setting of voltage regulation transformers using batch reinforcement learning, IEEE Transactions on Power Systems 35 (3) (2019) 1990–2001.

[29] D. Cao, J. Zhao, W. Hu, F. Ding, N. Yu, Q. Huang, Z. Chen, Model-free voltage control of active distribution system with PVs using surrogate model-based deep reinforcement learning, Applied Energy 306 (2022) 117982. doi:https://doi.org/10.1016/j.apenergy.2021.117982. URL https://www.sciencedirect.com/science/article/pii/S030626192101285X

[30] W. Wang, N. Yu, Y. Gao, J. Shi, Safe off-policy deep reinforcement learning algorithm for Volt-VAR control in power distribution systems, IEEE Transactions on Smart Grid 11 (4) (2020) 3008–3018. doi:10.1109/TSG.2019.2962625.

[31] F. Ruelens, B. J. Claessens, S. Vandael, B. De Schutter, R. Babuška, R. Belmans, Residential demand response of thermostatically controlled loads using batch reinforcement learning, IEEE Transactions on Smart Grid 8 (5) (2017) 2149–2159.

[32] L. R. Ferreira, A. R. Aoki, G. Lambert-Torres, A reinforcement learning approach to solve service restoration and load management simultaneously for distribution networks, IEEE Access 7 (2019) 145978–145987.

[33] M. Glavic, D. Ernst, L. Wehenkel, Combining a stability and a performance-oriented control in power systems, IEEE Transactions on Power Systems 20 (1) (2005) 525–526. doi:10.1109/TPWRS.2004.841146.

[34] T. L. Vu, S. Mukherjee, R. Huang, Q. Hung, Barrier function-based safe reinforcement learning for emergency control of power systems, arXiv preprint arXiv:2103.14186 (2021).

[35] Z. Yan, Y. Xu, Real-time optimal power flow: A Lagrangian based deep reinforcement learning approach, IEEE Transactions on Power Systems 35 (4) (2020) 3270–3273. doi:10.1109/TPWRS.2020.2987292.

[36] C. M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.

[37] I. Clavera, Y. Fu, P. Abbeel, Model-augmented actor-critic: Backpropagating through paths, in: International Conference on Learning Representations, 2019.

[38] T.-H. Pham, G. De Magistris, R. Tachibana, Optlayer-practical constrained optimization for deep reinforcement learning in the real world, in: 2018 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2018, pp. 6236–6243.

[39] G. Dalal, K. Dvijotham, M. Vecerik, T. Hester, C. Paduraru, Y. Tassa, Safe exploration in continuous action spaces, arXiv preprint arXiv:1801.08757 (2018).

[40] T. Cover, J. Thomas, Elements of Information Theory, Wiley, 2012.

[41] F. Leibfried, J. Grau-Moya, Mutual-information regularization in markov decision processes and actor-critic learning, in: Conference on Robot Learning, PMLR, 2020, pp. 360–373.

[42] M. I. Belghazi, A. Baratin, S. Rajeswar, S. Ozair, Y. Bengio, A. Courville, R. D. Hjelm, Mine: mutual information neural estimation, arXiv preprint arXiv:1801.04062 (2018).

[43] M. Janner, J. Fu, M. Zhang, S. Levine, When to trust your model: Model-based policy optimization, in: Advances in Neural Information Processing Systems, 2019, pp. 12519–12530.

[44] K. Chua, R. Calandra, R. McAllister, S. Levine, Deep reinforcement learning in a handful of trials using probabilistic dynamics models, arXiv preprint arXiv:1805.12114 (2018).

[45] W. H. Kersting, Radial distribution test feeders, in: IEEE Power Engineering Society Winter Meeting, Vol. 2, 2001, pp. 908–912.

[46] UK Power Networks, Smart meter energy consumption data in London households, https://data.london.gov.uk/dataset/smartmeter-energy-use-data-in-london-households.

[47] T. Haarnoja, A. Zhou, P. Abbeel, S. Levine, Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, in: International conference on machine learning, PMLR, 2018, pp. 1861–1870.

[48] G. Dulac-Arnold, D. Mankowitz, T. Hester, Challenges of real-world reinforcement learning, arXiv preprint arXiv:1904.12901 (2019).

[49] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, Pytorch: An imperative style, high-performance deep learning library, in: H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, R. Garnett (Eds.), Advances in Neural Information Processing Systems 32, Curran Associates, Inc., 2019, pp. 8024–8035.
URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[50] J. Löfberg, Yalmip : A toolbox for modeling and optimization in matlab, in: In Proceedings of the CACSD Conference, Taipei, Taiwan, 2004.

[51] Z. Wang, J. Wang, B. Chen, M. M. Begovic, Y. He, MPC-based Voltage/Var optimization for distribution circuits with distributed generators and exponential load models, IEEE Transactions on Smart Grid 5 (5) (2014) 2412–2420. doi:10.1109/TSG.2014.2329842.

Table 1: Hyperparameters

All	replay buffer size	4000
	discount factor γ	$\{0.95, 0.95, 0.99\}$
	number of hidden layers	2
	optimizer	Adam
	pre-train steps	500
SAC	reward scale	5
	temperature parameter α	$\{0.4, 0.4, 0.3\}$
	learning rate	0.001
	number of hidden units	$\{80, 90, 110\}$
	hidden activation (actor-critic)	tanh-tanh
	smoothing parameter ρ	0.99
	minibatch size	32
MAAC	reward scale	5
	model rollout length H	$\{16, 16, 13\}$
	number of ensemble N	10
	learning rate	0.001
	number of hidden units (actor-crtic)	$\{80, 80, 130\}$
	number of hidden units (model)	$\{80, 100, 100\}$
	hidden activation (actor-crtic-model)	tanh-relu-relu
	minibatch size	16
	exploration noise Std σ_ξ	0.1
SAAC	reward scale	$\{5, 5, 10\}$
	model rollout length H	$\{6, 6, 3\}$
	number of ensemble N	10
	learning rate	0.001
	number of hidden units (actor-crtic)	$\{80, 90, 130\}$
	number of hidden units (model)	$\{80, 100, 100\}$
	hidden activation (actor-crtic-model)	relu-relu-relu
	minibatch size	32
	exploration noise Std σ_ξ	0.1
	penalty coefficient sequence σ	[10., 10., 100., 100., 1000., 1000.]
	regularization constant β	1.0

Table 2: Asymptotic Performance Comparisons between the Proposed Method and Model-Based Methods

	Negative cumulative reward ($)			Voltage violation (p.u.)			Switching number		
System	4	34	123	4	34	123	4	34	123
SAC [47]	76.26	65.24	85.25	0.00	0.00	0.00	5	0	0
MAAC [37]	77.69	51.35	91.79	0.00	0.00	0.00	11	20	22
SAAC (ours)	75.82	54.38	76.03	0.00	0.00	0.00	2	24	16
MICP [8]	78.09	56.91	73.55	0.00	0.00	0.00	1	50	16
MPC [51]	75.20	52.98	72.81	0.00	0.00	0.00	9	74	17