Electrical Power and Energy Systems 142 (2022) 108265

Contents lists available at ScienceDirect

International Journal of Electrical Power and Energy Systems

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

Low latency cyberattack detection in smart grids with deep reinforcement learning☆

Yaze Li [*,1], Jingxian Wu[2]

Department of Electrical Engineering, University of Arkansas, Fayetteville, AR 72701, USA

ARTICLE INFO

Keywords:
Cybersecurity
Smart grid
False data injection (FDI)
Denial of service (DoS)
Dynamic state estimation (DSE)
Deep reinforcement learning (DRL)
Deep Q-network (DQN)

ABSTRACT

This paper focuses on low latency detection of cyberattacks in smart grids with deep reinforcement learning (DRL). The objective of low latency detection is to minimize detection delay while ensuring high detection accuracy. This is different from conventional detection methods that focus mainly on detection accuracy and pay little attention to detection delay. A lower detection delay can reduce recovery time, thus minimizing service interruption or economic losses due to cyberattacks. Since detection delay is the main design metric, the algorithm is developed by using a non-linear dynamic AC system model with an extended Kalman filter (EKF) to capture power grid state transitions in real-time, while many other works in the literature use a simplified linear DC model. The DRL detection algorithm is developed by using a continuous state space deep Q-network (DQN) on the framework of a Markov decision process (MDP). The new DQN design has two main innovations. First, the MDP state is designed as a sliding window of Rao-statistics of the AC dynamic state estimation residues. The proposed state formulation can accurately capture dynamic power state transitions in real-time. Second, a new reward function is designed to allow a flexible trade-off between detection delays and detection accuracy. The delay-accuracy trade-off can be adjusted by tuning a single parameter in the reward function. Simulation results show that the proposed DQN-based DRL detection algorithm can achieve very low detection delays with high detection accuracy.

1. Introduction

The ever-increasing power demands along with growing penetration rates of renewable energy necessitate designs of smart grids that are reliable, resilient, and secure [1]. An important component of a smart grid is the supervisory control and data acquisition (SCADA) system, which monitors and controls power grid operations with the help of remote terminal units (RTUs). However, SCADA systems are prone to cyberattacks. For instance, cyberattacks on the SCADA system in the power grid of Kiev, Ukraine on December 23, 2015 led to a wide range blackout [2].

There are various types of cyberattacks on smart grids. One of the most common cyberattacks is false data injection (FDI) attack, which can target different layers and systems in the smart grid [3]. A classic FDI attack is designed to change some SCADA measurements to decrease the accuracy of state estimation (SE), which results in unreliable power system operations [4]. Another type of cyberattack is denial-of-service (DoS) attack. The DoS attack can negatively impact SCADA system operations by blocking the communication links among devices or making some measurement devices unavailable [5].

A plethora of algorithms have been developed for cyberattack detection by using power system state estimations, where the measurement results can be compared to estimation results to identify the presence of anomalies. Many algorithms are developed by using static state estimation (SSE) with a simplified DC system model due to its low computational complexity [6–8]. However, SSE cannot capture the dynamic state transitions in power systems, and it is in general not suitable for real-time monitoring of power system operations. Dynamic state estimation (DSE) with Kalman filter (KF) and its derived algorithms are widely used for power system estimation and intrusion detection [9]. A KF-based DSE algorithm is used to detect FDI in automatic generation control (AGC) system in [10]. Distributed Kalman filter (DKF) was used in [11] to reduce the computation complexity for attack detection, and

☆ The work was supported in part by the U.S. National Science Foundation (NSF) under Grant ECCS-1711087 and U.S. Department of Energy under Award Number DE-OE0000779.

* Corresponding author.
E-mail address: yazeli@uark.edu (Y. Li).
[1] Student Member, IEEE.
[2] Senior Member, IEEE.

https://doi.org/10.1016/j.ijepes.2022.108265
Received 18 January 2022; Received in revised form 11 March 2022; Accepted 11 April 2022
Available online 6 May 2022
0142-0615/Published by Elsevier Ltd.

extended Kalman filter (EKF) was used in [12,13] to model the nonlinear measurement function of AC power system model for FDI attack detection. A robust Cubature Kalman Filter (RCKF) based approach is proposed for systems with power generators under cyberattacks [14]. In [15], a correlation-based DSE is proposed to detect DoS attacks.

With the recent rapid advances in artificial intelligence (AI) and machine learning (ML), there have been growing interests in applying ML algorithms for intrusion or cyberattack detection in smart grids. In [16], a real time FDI detection algorithm is developed by performing robust principal component analysis (PCA). Various deep learning algorithms, such as deep belief network (DBN) [17,18], recurrent neural network (RNN) [19], and deep neural network (DNN) [20], are developed for FDI detection. Reservoir computing (RC) is an extension framework of NN. It consists of three parts: a feed-forward NN as the input layer, an RNN as the middle layer, and a weighted adder as the output layer [21]. A delayed feedback RC is used for detecting dynamic attacks in smart grids [22,23]. These algorithms utilize a data-driven approach, that is, the input to the neural networks are measurements from the power system, and the output are detection results. Preprocessing of the measurements can be done to make the training more efficient, such as the wavelet transform in [24]. A state–action–reward–state–action (SARSA) algorithm based on reinforcement learning (RL) is developed for FDI detection in [25]. A Q-Learning method with nearest sequence memory is adopted to detect FDI attack to automatic voltage control (AVC), where the values of the Q-function are discretized and stored in a lookup table [26]. In [27], a deep Q-network (DQN) is designed to defend FDI in power grids, and it utilizes a DNN with discrete states to approximate the Q-function required for Q-learning.

Despite a large number of research works on cyberattack detection in smart grids, most existing methods aim at improving detection accuracy yet paid little or no attention to detection delay. Detection delay is critical for sustaining the reliability and security of power grids. A low detection delay can ensure the timely recovery from cyberattacks, which can provide uninterrupted grid operations and minimize economic losses. Sequential quickest change detection algorithms such as cumulative sum (CUSUM) [28] can minimize the detection delay. However, the CUSUM algorithm requires perfect knowledge of the statistical distributions of the data before and after the attack, and this information is usually not readily available in practical systems.

We propose to address this problem by developing a low latency detection algorithm that aims at minimizing the detection delay while maintaining good detection accuracy. The proposed algorithm adopts a hybrid model- and data-driven approach that relies on both the physical model of the power grid and the measurement data collected from the grid. In the model-based analysis, an AC model with an extended Kalman filter (EKF) is used to estimate and track the dynamic transitions of the power system. The data-driven analysis is performed by developing a deep reinforcement learning (DRL) based detection algorithm with a deep Q-network (DQN) on the framework of the Markov decision process (MDP). The new DQN design has two main innovations. First, the MDP state is designed as a sliding window of the Rao-statistics of the AC dynamic state estimation residuals. Such a state representation can accurately capture the dynamic state transitions in power systems over certain time periods, thus enabling real time detection. Second, a new reward function is proposed to enable flexible trade-offs between the detection delay and detection accuracy. The combination of the new MDP state and reward function allows us to achieve low latency attack detection in real time with high detection accuracy, and it can be used to detect both FDI and DoS attacks. In addition, the proposed DQN algorithm utilizes a continuous state space instead of the discrete state space used by most existing RL algorithms. The adoption of continuous state space can reduce detection complexity and improve detection accuracy, and it makes the algorithm less likely to suffer from the curse of dimensionality.

The rest of this article is organized as follows. The AC system model of a smart grid is introduced in Section 2. The dynamic state estimation of the power grid is described in Sections Section 3. The problem formulation along with the models of both FDI and DoS attacks are given in 4. Section 5 provides the proposed low latency detection algorithm with DQN. Simulation results are presented in Section 6, and Section 7 concludes the paper.

2. System model

We consider a power system with N buses. Without loss of generality, the first bus is chosen to be the slack (reference) bus, which means the phase of the voltage at this bus is regarded as 0. The magnitudes and phases of voltages on the $N-1$ remaining buses are states of the system. Define the state vector of the system as $\mathbf{x} = [x_1, x_2, \dots, x_n]^T \in \mathcal{D}^{n\times 1}$, where $n = 2N-1$, and $\mathbf{A}^T$ is the matrix transpose operator. Denote the active and reactive power injected to bus i as P_i and Q_i, respectively, and the number of buses that are connected to bus i is c_i. Denote the active and reactive power flow from bus i to bus j as P_{ij} and Q_{ij}, respectively.

Each bus is equipped with a smart meter that collects the active and reactive power injections and power flows. The system provides $m = m_1 + m_2 + 1$ measurements in total, where $m_1 = 2N$ is the number of active and reactive power injections, and $m_2 = \sum_{i=1}^{N} c_i$ is the number of active and reactive power flows. To fully observe the power system, the voltage magnitude at the slack bus V_1 is also collected. Denote the measurement vector as $\mathbf{z} = [z_1, z_2, \dots, z_m]^T \in \mathcal{D}^{m\times 1}$. The power and voltage results at time t can be modeled as nonlinear functions of the state vector $\mathbf{x}_t$ as $\mathbf{h}(\mathbf{x}_t) = [h_1(\mathbf{x}_t), h_2(\mathbf{x}_t), \dots, h_m(\mathbf{x}_t)]^T$. The measurement vector can then be represented by

$$\mathbf{z}_t = \mathbf{h}(\mathbf{x}_t) + \mathbf{e}_t, \tag{1}$$

where $\mathbf{e}_t$ is the measurement error at time t, and it is modeled with a zero-mean Gaussian distributed random vector with length m and covariance matrix $\mathbf{R}$. Denote the estimated state vector with a dynamic state estimation at time t as $\hat{\mathbf{x}}_t$. Define a residual vector $\mathbf{v}_t$ as

$$\mathbf{v}_t = \mathbf{z}_t - \mathbf{h}(\hat{\mathbf{x}}_t). \tag{2}$$

3. Dynamic state estimation

There are various DSE methods used for power system estimation based on Kalman filters (KF). Most follow a two-step structure: state prediction followed by state estimation, but with different prediction models and KF models. In this section, we present a DSE using Holt's linear trend method for state prediction, and extended Kalman filter (EKF) for state estimation.

3.1. Holt's linear trend method

The state vector $\{\mathbf{x}_t\}_t$ is a non-linear time series. We propose to model the state vector by using Holt's linear trend method, which uses exponential smoothing to forecast a non-linear dynamic time series with a trend [29]. It involves a forecast equation and two smoothing constants, α and β.

Consider a dynamic time series $\{y_t\}_t$. The h-step forecast equation for the time series is

$$\tilde{y}_{t+h|t} = l_t + hb_t \tag{3}$$

where $\tilde{y}_{t+h|t}$ denotes the h-step forecast from y_t, and l_t and b_t are the estimations of the level and trend (slope) of the series at time t, respectively. The values of l_t and b_t are iteratively updated as

$$l_t = \alpha y_t + (1-\alpha)\tilde{y}_{t|t-1} = \alpha y_t + (1-\alpha)(l_{t-1} + b_{t-1}) \tag{4}$$

$$b_t = \beta(l_t - l_{t-1}) + (1-\beta)b_{t-1} \tag{5}$$

where both α and β are smoothing parameters between 0 and 1. The level Eq. (4) shows that l_t is a weighted average of observation y_t and one-step-ahead forecast $\tilde{y}_{t|t-1}$. The trend Eq. (5) shows that b_t is a

weighted average of the estimated trend and the first order difference of the estimated level. The initial values l_0 and b_0 are estimated by minimizing the sum of squared errors for the one-step training errors. Since the level and trend are updated for each t, the forecasting method is dynamic. Applying (3)–(5) with $h = 1$ to the power system state vector $\mathbf{x}_t$, we can express the forecast state vector $\tilde{\mathbf{x}}_{t+1}$ from $\mathbf{x}_t$ as

$$\tilde{\mathbf{x}}_{t+1} = \mathbf{F}_t \mathbf{x}_t + \mathbf{g}_t \tag{6}$$

where

$$\mathbf{F}_t = \alpha(1+\beta)\mathbf{I}_n, \tag{7}$$

$$\mathbf{g}_t = (1+\beta)(1-\alpha)\tilde{\mathbf{x}}_t - \beta \mathbf{l}_{t-1} + (1-\beta)\mathbf{b}_{t-1} \tag{8}$$

In the above equations, $\mathbf{I}_n$ is a size-n identity matrix, $\mathbf{l}_t$ and $\mathbf{b}_t$ are the estimates of the level and trend vectors, respectively.

With the predictive model given in (6), The dynamic transition of the state vector $\mathbf{x}$ in the power system can be modeled as

$$\mathbf{x}_{t+1} = \mathbf{F}_t \mathbf{x}_t + \mathbf{g}_t + \mathbf{w}_t \tag{9}$$

where $\mathbf{F}_t \in \mathcal{D}^{n\times n}$ is a diagonal matrix, $\mathbf{g}_t \in \mathcal{D}^{n\times 1}$ is a non-zero column vector, and $\mathbf{w}_t$ is a white Gaussian noise vector with zero mean and covariance matrix $\mathbf{Q}_t$, accounting for model uncertainties.

In practical systems the true values of the state vector $\mathbf{x}_t$ is unknown, and a state estimation $\hat{\mathbf{x}}_t$ is utilized. Denote the error covariance matrix of $\hat{\mathbf{x}}_t$ as $\mathbf{\Sigma}_t = \mathbb{E}\left[(\mathbf{x}_t - \hat{\mathbf{x}}_t)(\mathbf{x}_t - \hat{\mathbf{x}}_t)^T\right]$, where $\mathbb{E}[\cdot]$ is the expectation operator.

The state forecasting at time $t+1$ from the estimated state $\hat{\mathbf{x}}_t$ can be written as

$$\tilde{\mathbf{x}}_{t+1} = \mathbb{E}\left[\mathbf{x}_{t+1} | \mathbf{x}_t = \hat{\mathbf{x}}_t\right] = \mathbf{F}_t \hat{\mathbf{x}}_t + \mathbf{g}_t \tag{10}$$

The corresponding error covariance matrix of state forecasting can then be calculated as

$$\begin{aligned}\mathbf{M}_{t+1} &= \mathbb{E}\left[\left(\mathbf{x}_{t+1} - \tilde{\mathbf{x}}_{t+1}\right)\left(\mathbf{x}_{t+1} - \tilde{\mathbf{x}}_{t+1}\right)^T | \mathbf{x}_t = \hat{\mathbf{x}}_t\right] \\ &= \mathbf{F}_t \mathbf{\Sigma}_t \mathbf{F}_t + \mathbf{Q}_t.\end{aligned} \tag{11}$$

3.2. State estimation

The state estimation at time $t+1$, denoted as $\hat{\mathbf{x}}_{t+1}$, can be obtained by minimizing the following objective function,

$$\begin{aligned}J\left(\mathbf{x}_{t+1}\right) = &\frac{1}{2}\left[\mathbf{z}_{t+1} - \mathbf{h}(\mathbf{x}_{t+1})\right]^T \mathbf{R}_{t+1}^{-1}\left[\mathbf{z}_{t+1} - \mathbf{h}(\mathbf{x}_{t+1})\right] \\ &+ \frac{1}{2}\left[\left(\mathbf{x}_{t+1} - \tilde{\mathbf{x}}_{t+1}\right)^T \mathbf{M}_{t+1}^{-1}\left(\mathbf{x}_{t+1} - \tilde{\mathbf{x}}_{t+1}\right)\right],\end{aligned} \tag{12}$$

where $\tilde{\mathbf{x}}_{t+1}$ is the forecast state vector in (10), and $\mathbf{z}_{t+1}$ is the newly received measurement at time $t+1$.

The optimum $\hat{\mathbf{x}}_{t+1}$ that minimizes $J\left(\mathbf{x}_{t+1}\right)$ can be obtained through an iterative EKF as [30]

$$\begin{aligned}\hat{\mathbf{x}}^{(i+1)} = &\hat{\mathbf{x}}^{(i)} + \mathbf{\Sigma}^{(i)}\{\mathbf{H}^T(\hat{\mathbf{x}}^{(i)})\mathbf{R}^{-1}[\mathbf{z} - \mathbf{h}(\hat{\mathbf{x}}^{(i)})] \\ &- \mathbf{M}^{-1}[\hat{\mathbf{x}}^{(i)} - \tilde{\mathbf{x}}]\},\end{aligned} \tag{13}$$

where i denotes the iteration counter, $\mathbf{H}(\mathbf{x}) = \frac{\partial \mathbf{h}(\mathbf{x})}{\partial \mathbf{x}}$ is the Jacobian matrix, and $\mathbf{\Sigma}^{(i)}$ is the error covariance matrix of the estimation $\hat{\mathbf{x}}^{(i)}$ as

$$\mathbf{\Sigma}^{(i)} = \left[\mathbf{H}^T(\hat{\mathbf{x}}^{(i)})\mathbf{R}^{-1}\mathbf{H}(\hat{\mathbf{x}}^{(i)}) + \mathbf{M}^{-1}\right]^{-1}. \tag{14}$$

It should be noted that the subscript $t+1$ was omitted in (13) and (14) for simplicity.

The EKF is initialized with $\hat{\mathbf{x}}_{t+1}^{(0)} = \tilde{\mathbf{x}}_{t+1}$. Under the assumption that state forecasting has a high accuracy, that is, $|\mathbf{x}_{t+1} - \tilde{\mathbf{x}}_{t+1}|$ is very small, the EKF initialized with $\tilde{\mathbf{x}}_{t+1}$ will converge very fast. Thus we only consider the estimation result after one EKF iteration. Performing the iteration in (13) and (14) once yields

$$\hat{\mathbf{x}}_{t+1} = \tilde{\mathbf{x}}_{t+1} + \mathbf{K}_{t+1}\mathbf{v}_{t+1}, \tag{15}$$

$$\mathbf{\Sigma}_{t+1} = \left[\mathbf{H}^T(\tilde{\mathbf{x}}_{t+1})\mathbf{R}_{t+1}^{-1}\mathbf{H}(\tilde{\mathbf{x}}_{t+1}) + \mathbf{M}_{t+1}^{-1}\right]^{-1}, \tag{16}$$

where $\mathbf{K}_{t+1}$ is the Kalman gain matrix defined as,

$$\mathbf{K}_{t+1} = \mathbf{\Sigma}_{t+1}\mathbf{H}^T(\tilde{\mathbf{x}}_{t+1})\mathbf{R}_{t+1}^{-1}, \tag{17}$$

and

$$\mathbf{v}_{t+1} = \mathbf{z}_{t+1} - \mathbf{h}(\tilde{\mathbf{x}}_{t+1}), \tag{18}$$

is the residual vector.

The results from DSK with EKF will be used to facilitate the design of low latency cyberattack detection algorithms.

4. Problem formulation

Two types of attacks are considered in this paper: false data injection (FDI) attack, and denial of service attack (DoS). The detection of these two types of attacks can be formulated as a hypothesis test.

4.1. Attack models

Assume that measurement elements in index $\mathcal{I} \subseteq \{1, 2, \ldots, m\}$ are attacked at time τ. The attack models of FDI and DoS are given as follows.

(1) False Data Injection (FDI): The measurement vector is injected with a random attack vector $\mathbf{a} = [a_1, a_2, \ldots, a_m]^T \in \mathcal{D}^{m\times 1}$, with $a_i = 0$ if $i \notin \mathcal{I}$:

$$\mathbf{z}_t = \begin{cases} \mathbf{h}(\mathbf{x}_t) + \mathbf{e}_t, & t < \tau \\ \mathbf{h}(\mathbf{x}_t) + \mathbf{e}_t + \mathbf{a}, & t \geq \tau \end{cases} \tag{19}$$

(2) Denial of Service (DoS): In the DoS attack, a subset of the elements in the measurement vector are changed to zero. The DoS attack can be represented by using a diagonal matrix $\mathbf{A}$ with the main diagonal being a binary vector $\mathbf{d} = [d_1, d_2, \ldots, d_m]^T \in \{0, 1\}^m$, that is $\mathbf{A} = \text{diag}(\mathbf{d})$, where

$$d_i = \begin{cases} 0, & i \in \mathcal{I} \\ 1, & i \notin \mathcal{I} \end{cases} \tag{20}$$

The DoS attack is modeled as:

$$\mathbf{z}_t = \begin{cases} \mathbf{h}(\mathbf{x}_t) + \mathbf{e}_t, & t < \tau \\ \mathbf{A}\left[\mathbf{h}(\mathbf{x}_t) + \mathbf{e}_t\right], & t \geq \tau \end{cases} \tag{21}$$

DoS attack can happen on different layers in the smart grid. Lack of measurements might cause the system to shut down in some cases.

In this paper the DoS attack is modeled by setting the unavailable measurements as zero. In this case, the DoS attack can be considered as a special case of FDI attack, because the DoS attack on $\mathcal{I}$ is equivalent to an FDI attack with an attack vector:

$$a_i = \begin{cases} -z_i, & i \in \mathcal{I} \\ 0, & i \notin \mathcal{I} \end{cases} \tag{22}$$

Thus model (19) will be used in this paper for both attacks.

4.2. Hypothesis test

The attack detection problem can be formulated in the form of a hypothesis test, where the null and alternate hypotheses correspond to the status of normal operation and attack, respectively.

Define the null hypothesis $\mathcal{H}_0$ and alternate hypothesis $\mathcal{H}_1$ at time $t+1$ as

$$\begin{aligned}\mathcal{H}_0 &: \mathbf{z}_{t+1} = \mathbf{h}(\mathbf{x}_{t+1}) + \mathbf{e}_{t+1}, \\ \mathcal{H}_1 &: \mathbf{z}_{t+1} = \mathbf{h}(\mathbf{x}_{t+1}) + \mathbf{e}_{t+1} + \mathbf{a}.\end{aligned} \tag{23}$$

Based on the assumption of high forecast accuracy, the nonlinear measurement function $\mathbf{h}(\mathbf{x}_{t+1})$ at time $t+1$ can be approximated by using its first order Taylor series expansion around point $\tilde{\mathbf{x}}_{t+1}$ as,

$$\mathbf{h}(\mathbf{x}_{t+1}) = \mathbf{h}(\tilde{\mathbf{x}}_{t+1}) + \mathbf{H}(\tilde{\mathbf{x}}_{t+1})(\mathbf{x}_{t+1} - \tilde{\mathbf{x}}_{t+1}). \tag{24}$$

Combining (18), (19), and (24) yields

$$\mathbf{v}_{t+1} = \mathbf{H}(\tilde{\mathbf{x}}_{t+1})(\mathbf{x}_{t+1} - \tilde{\mathbf{x}}_{t+1}) + \mathbf{e}_{t+1}. \tag{25}$$

The covariance matrix of $\mathbf{v}_{t+1}$ in the above equation is

$$\mathbf{S}_{t+1} = \mathbf{H}(\tilde{\mathbf{x}}_{t+1})\mathbf{M}_{t+1}\mathbf{H}^T(\tilde{\mathbf{x}}_{t+1}) + \mathbf{R}_{t+1}. \tag{26}$$

The residual vector after attack can be obtained in a similar manner. Then the hypothesis test given in (23) can be equivalently expressed in the form of the residual vector $\mathbf{v}_{t+1}$ as

$$\begin{aligned} \mathcal{H}_0 &: \mathbf{v}_{t+1} = \mathbf{H}(\tilde{\mathbf{x}}_{t+1})(\mathbf{x}_{t+1} - \tilde{\mathbf{x}}_{t+1}) + \mathbf{e}_{t+1}, \\ \mathcal{H}_1 &: \mathbf{v}_{t+1} = \mathbf{H}(\tilde{\mathbf{x}}_{t+1})(\mathbf{x}_{t+1} - \tilde{\mathbf{x}}_{t+1}) + \mathbf{e}_{t+1} + \mathbf{a}. \end{aligned} \tag{27}$$

The residual $\mathbf{v}_{t+1}$ is assumed to be Gaussian distributed, with its mean vector being zero and $\mathbf{a}$ under the null and alternate hypothesis, respectively [31]. The covariance matrix remains $\mathbf{S}_{t+1}$ with or without attacks. Thus the hypothesis test can be equivalently written as

$$\begin{aligned} \mathcal{H}_0 &: \mathbf{v}_{t+1} \sim \mathcal{N}(\mathbf{0}, \mathbf{S}_{t+1}), \\ \mathcal{H}_1 &: \mathbf{v}_{t+1} \sim \mathcal{N}(\mathbf{a}, \mathbf{S}_{t+1}). \end{aligned} \tag{28}$$

To further simplify the detection process, we perform the eigen-decomposition of $\mathbf{S}_{t+1}$ as

$$\mathbf{S}_{t+1} = \mathbf{U}_{t+1}^T \mathbf{D}_{t+1} \mathbf{U}_{t+1}. \tag{29}$$

Define a whitened residual vector

$$\bar{\mathbf{v}}_{t+1} = \mathbf{W}_{t+1}\mathbf{v}_{t+1} \tag{30}$$

where $\mathbf{W}_{t+1} = \mathbf{D}_{t+1}^{-\frac{1}{2}}\mathbf{U}_{t+1}$ is the whitening matrix. Then the hypothesis test on $\bar{\mathbf{v}}_{t+1}$ can be alternatively expressed as

$$\begin{aligned} \mathcal{H}_0 &: \bar{\mathbf{v}}_{t+1} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_m), \\ \mathcal{H}_1 &: \bar{\mathbf{v}}_{t+1} \sim \mathcal{N}(\mathbf{W}_{t+1}\mathbf{a}, \mathbf{I}_m). \end{aligned} \tag{31}$$

5. Quickest attack detection with DQN

Based on the hypothesis test formulated in the previous section, the proposed quickest attack detection method with DQN is presented in this section.

In quickest attack detection, the objective is to minimize the average detection delay (ADD) subject to an upper bound of the probability of false alarm (PFA). Denote the attack time identified by the detector as $\hat{\tau}$. Then the ADD and PFA can be evaluated as

$$\text{ADD} = \mathbb{E}[\hat{\tau} - \tau | \hat{\tau} > \tau] \tag{32}$$

$$\text{PFA} = \text{P}(\hat{\tau} < \tau) \tag{33}$$

The design of optimum quickest attack detection algorithm usually requires the knowledge of the attack vector $\mathbf{a}$, which is not available in practical systems. We propose to solve this problem by using the Rao-test statistic of the whitened residual as [32,33]

$$Y_t = \bar{\mathbf{v}}_t^T \bar{\mathbf{v}}_t \tag{34}$$

With the Rao-test statistic given in (34), we propose to model the system as a MDP and solve it by a using an RL algorithm.

5.1. MDP formulation

The MDP is a tuple of $(O, A, T, r, \Omega, \gamma)$, where O is the state space, A is the action space, T is the conditional transition probability between states, r is the reward function, Ω is the conditional transition probability between observations, and $\gamma \in [0, 1]$ is the reward discount factor [34]. Next we will formulate the attack detection problem into the MDP framework.

- State

  The state of the system should be able to reveal the status change of the power grid before and after an attack. The system measurement vector $\mathbf{z}_t$ is in general not a good candidate for state, because it is possible to obtain the same measurement before and after a carefully designed attack. Various system operation and measurement statistics have been used as state in the literature, such as the state estimation residual [27], the negative log-likelihood function of the DC dynamic state estimation [25], etc. Motivated by the Rao-CUSUM method presented in [33], we propose to represent the state at time t by using a size-w sliding window of Rao-test statistics as

  $$o(t) = [Y_{t-w+1}, Y_{t-w+2}, \dots, Y_t]. \tag{35}$$

  The state at time t contains the Rao-test statistics calculated from the current and the past $w-1$ measurements. The time evolution of the state can then be used to detect the presence of attacks.

- Action

  Since the objective of the system is attack detection, the action at time t can be simply defined as:

  $$a(t) = \begin{cases} 1, & \text{attack detected} \\ 0, & \text{no attack detected} \end{cases} \tag{36}$$

- Reward function

  Denote the reward function of taking action $a(t)$ from state $o(t)$ as $r(o(t), a(t))$. The design of the reward function plays an important role in the accuracy, efficiency, and convergence of the learning algorithm. The reward function should take into considerations of both detection accuracy and detection delay. The function will give rewards to correct detection and low detection delays, while false alarms and long detection delays should be penalized. Considering both detection accuracy and detection delay, we propose a new reward function as follows

  $$r(o(t), a(t)) = \begin{cases} \frac{1}{\tau}[1 - a(t)], & t < \tau, \\ \frac{1}{\tau}a(t), & t = \tau, \\ -\phi[1 - a(t)], & t > \tau, \end{cases} \tag{37}$$

  where $\phi \in [0, 1]$ is a parameter adjusting the trade-off between ADD and PFA. If ϕ is close to 0, then there is a very small penalty to delayed detection, which leads to a long ADD but a low PFA. On the other hand, a larger ϕ will lead to a severe penalty to long detection delays, which results in low ADD at the cost of potentially higher PFA. The stage transitions related to various actions and the corresponding reward functions are illustrated in Fig. 1.

  Assume the attack is detected at time $\hat{\tau}$, that is, $a(t) = 0$ for $t < \hat{\tau}$ and $a(\hat{\tau}) = 1$. Define an episode $\mathcal{E}$ as a sequence of state and action pairs between $t \in \{1, 2, \dots, \hat{\tau}\}$, that is, $\mathcal{E} = \{(o(1), a(1)), (o(2), a(2)), \dots, (o(\hat{\tau}), a(\hat{\tau}))\}$. Then the accumulated episode reward function is

  $$r(\mathcal{E}) = \sum_{t=1}^{\hat{\tau}} r(o(t), a(t)). \tag{38}$$

  An action of $a = 1$ will always lead to the terminal stage, that is, the end of an episode. The reward function in (37) is designed to have an accumulated episode reward of 1 for perfect detection, that is, $r(\mathcal{E}) = 1$ if $a(\tau) = 1$. The reward function will be strictly less than 1 for false alarm or delayed detection. For excessive long delays, the reward function will be negative. In (38), the accumulated episode reward will be negative if the detection delay is larger than $\frac{\tau - 1}{\phi\tau}$.

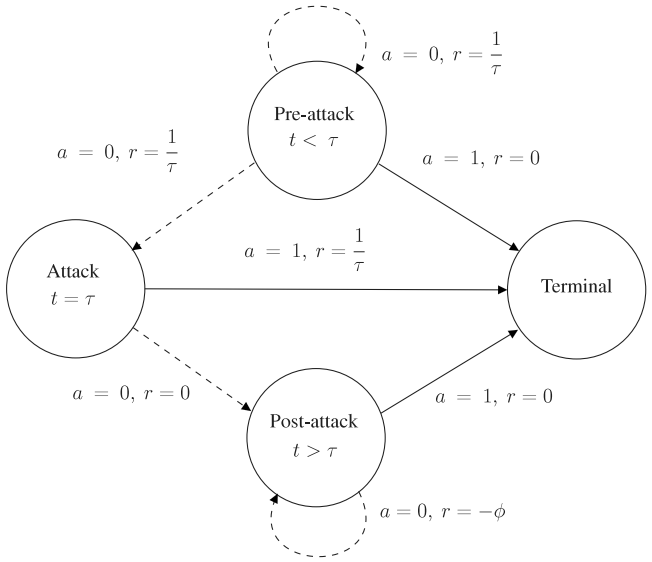

Fig. 1. Stage transitions.

5.2. DQN training

Based on the MDP formulation, we can formulate the quickest attack detection problem under the framework of DQN. In the proposed DQN method, we adopt a deterministic detection policy, that is, the action $a(t)$ is a deterministic function of the state $o(t)$ as $a(t) = \mu(o(t))$. Define the discounted cumulative future reward starting from t as

$$R_t = \sum_{i=t}^{\infty} \gamma^{i-t} r(o(i), a(i)) \tag{39}$$

where $\gamma \in (0, 1]$ is a discount factor, and the reward function $r(o(i), a(i))$ is defined in (37).

DQN and general Q-Learning methods use an action-value function, $Q(o(t), a(t))$, to estimate the expected cumulative future reward starting from t after taking action $a(t) = \mu(o(t))$ from the state $o(t)$ as

$$Q(o(t), a(t)) = \mathbb{E}_{r(i\geq t), o(i\geq t)}[R_t | o(t), \mu(o(t))] \tag{40}$$

where the expectations are performed with respect to future states $o(i \geq t)$ and future step rewards $r(i \geq t)$.

The action-value function can be written in a recursive form as the well-known Bellman equation as

$$\begin{aligned} Q(o(t), a(t)) &= r(o(t), a(t)) \\ &\quad + \gamma \mathbb{E}_{o(t+1)}[Q(o(t+1), \mu(o(t+1)))] \end{aligned} \tag{41}$$

If the optimal action-value function $Q^*(o, a)$ is known, then the optimal solution of Eq. (41) is a greedy policy

$$\mu^*(o) = \text{argmax}\, a \in \mathcal{A} Q^*(o, a), \tag{42}$$

where $\mathcal{A}$ is the action space, that is, the set containing all possible actions.

In practice, the Q-function is unknown. In Q-learning methods, the Q-function along with the policy are learned and updated in an iterative manner.

For a given Q-function at time t, the policy used in Q-learning is the greedy policy that chooses the best estimated cumulative future reward,

$$\mu(o(t)) = \text{argmax}\, a(t) \in \mathcal{A} Q(o(t), a(t)). \tag{43}$$

The Q function at time t is updated with a learning rate $\alpha \in [0, 1]$ as

$$\begin{aligned} Q(o(t), a(t)) \leftarrow &(1-\alpha) Q(o(t), a(t)) + \alpha [r(o(t), a(t)) \\ &+ \gamma \max_{a(t+1)\in\mathcal{A}} Q(o(t+1), a(t+1))]. \end{aligned} \tag{44}$$

In conventional Q-learning methods, the Q-function can be updated in a Q-table that stores all state–action pairs. The size of the table becomes prohibitively large when the state and action spaces are large. DQN replaces the Q-table with a neural network, Q-network, which approximates the Q-function by updating the weights θ in the neural network. The inputs to the Q-network are $o(t)$ and $a(t)$ and the output is $Q(o(t), a(t))$.

Due to the trade-off between exploration and exploitation, a direct implementation of (43) and (44) in the iterative Q-learning process might not lead to the desired results, mainly due to the fact that the learning environment is not sufficiently explored with the simple greedy policy in (43). In order to improve the stability and convergence rate of the learning process, we adopt several modifications to DQN, including ϵ-greedy policy in action selection, target network, and replay buffer with mini-batch [35]. Details of the proposed DQN learning process are given as follows.

Assume the entire DQN training time horizon is divided into multiple sequential training episodes. Each episode has at most T time steps. A training episode ends if an attack is detected before T time steps, or no attack is detected in T time steps. For each new episode, the power grid is initialized to the same normal operating condition, but with different random attack vectors applied at different time steps. The entire training process consists E training episodes.

In order to stabilize the training process and avoid big swings from step to step, we adopt a target network $Q'(o, a)$, which is built along the main network $Q(o, a)$. The weights of the main network, θ, are updated every step, yet the weights of the target network, θ', are copied from the main network every C steps, with C being an integer. The action selection is performed by using the target network $Q'(o, a)$.

In order to broaden the exploration of the action space, we adopt an ϵ-greedy policy in action selection. In the ϵ-greedy policy, a given probability parameter $\epsilon \in [0, 1]$ is chosen. During the action selection process, we either select a random action with probability ϵ, or greedily select an action by using (43) with probability $1-\epsilon$. Such a randomized action selection approach can broader the search space and avoid being trapped in a local optimum early during the training process.

Experience replay is used to improve the sample efficiency and stability of the learning process. In experience replay, we can obtain in each time step an experience tuple, $e(t) = \{o(t), a(t), r(t), o(t+1)\}$, which is stored in a replay buffer $\mathcal{D}$. The weights of the main network are updated by randomly sample a mini-batch $\mathcal{B} \subseteq \mathcal{D}$ experience tuples from the replay buffer. For each experience tuple in the mini-batch, we first calculate the corresponding target value as

$$y(i) = \begin{cases} r(i), & \text{if } i = T \\ r(i) + \gamma \max_{a\in\mathcal{A}} Q'(o(i+1), a; \theta'), & \text{otherwise} \end{cases} \tag{45}$$

Then the weights of the main network can be updated by minimizing the mean squared error between the main network and the target values of the mini-batch as

$$\min_{\theta} \frac{1}{|\mathcal{B}|} \sum_{e(i)\in\mathcal{B}} [y(i) - Q(o(i), a(i); \theta)]^2 \tag{46}$$

The minimization can be performed by using gradient descent. With experience replay, the main network weights are updated by using a random subset of experience tuples. Such an approach allows the algorithm to learn from uncorrelated experiences from the past, recall rare occurrences, and learn from individual experiences multiple times. As a result, it can learn more efficiently from the past experience with better stability.

Details of the DQN training procedure are shown in Algorithm 1.

Algorithm 1 DQN Learning

Require: Bus number N, measurement size m, window size w, attack time τ, initial action-value network parameter θ, empty replay buffer $\mathcal{D}$

Initialization: Set target network weights: $\theta' \leftarrow \theta$.
for $e = 1$ to E **do**
  Initialization: Set $t \leftarrow 1$; calculate $o(1)$ by dynamic state estimation.
  while $t \leq T$ **do**
    Select action $a(t)$ by following the ϵ-greedy policy.
    if $a(t) = 1$ **then**
      $t \leftarrow T + 1$
    else
      Calculate reward $r(t)$ by using (37).
      Calculate $o(t+1)$ by dynamic state estimation.
      Store experience tuple $(o(t), a(t), r(t), o(t+1))$ in the replay buffer $\mathcal{D}$.
      Randomly sample a mini-batch of $|\mathcal{B}|$ experience tuples $\mathcal{B} = \{(o(i), a(i), r(i), o(i+1))\}$ from $\mathcal{D}$.
      Calculate the target value $y(i)$ for all experience tuples in $\mathcal{B}$ by using (45).
      Update the main network weights θ by minimizing the cost function in (46).
      if $\text{mod}\ (t, C) = 0$ **then**
        Update the target network weights $\theta' \leftarrow \theta$.
      end if
      $t \leftarrow t + 1$
    end if
  end while
  $e \leftarrow e + 1$
end for

Ensure: Target network parameters θ'

5.3. Complexity analysis

The computation complexity of the proposed algorithm comes from two sources: the dynamic state estimation (DSE) with extended Kalman filter (EKF), and the DQN algorithm.

During DSE with an AC system model, the EKF is used to estimate and track the dynamic state transition of the power grid. The estimation results are then used to calculate the Rao-statistics to form the state vector for the DQN. This procedure is performed at each time step during the training and testing process.

Consider a power system with N buses and M lines. The size of the state vector $\mathbf{x}$ is $n \times 1$, where $n = 2N - 1$. The dimension of the measurement vector $\mathbf{z}$ is $m \times 1$, where $m = 2(M+N)+1$. The Holt's linear trend method in Section 3.1 requires 2 vector add (VA) in (8), and multiple scalar multiplications in (7) and (8). The computational cost of scalar multiplication is much lower than matrix operations such as VA, matrix–vector product (MVP), matrix–matrix product (MMP), and matrix inversion (MI). Therefore, only VA, MVP, MMP, MI are counted during the complexity analysis. The state forecasting in (10) has 1 VA and 1 MVP, the calculation of error covariance matrix in (11) has 2 MMPs and 1 VA.

The state estimation is obtained from one EKF iteration in (15), which has 1 MVP and 1 VA. The calculation of the residual vector requires 1 VA in (18). The Kalman gain matrix requires 2 MMPs in (17) and 2 MMPs, 1 VA, and 2 MI in (16). A summary of vector and matrix operations of the DSE-EKF is given in Table 1. Given $M > N$ in most cases, the DSE has a computation complexity of $\mathcal{O}(N(M+N)^2)$.

Regarding the DQN algorithm, we only need to consider the complexity of the online detection process, because the training is performed offline. The computational complexity of the online DQN algorithm comes from the computation cost in the target network. During the online DQN detection process, the target network takes an input $o(t)$ of dimension w. It has two hidden layers with dimension k each, and generates an output of dimension 2, $Q(o(t), 0; \theta')$ and $Q(o(t), 1; \theta')$. The target network has 4 layers with a total of $w + 2k + 2$ $(w, k, k, 2)$ neurons. For each layer, an MVP and an activation function are computed. Thus the online DQN detection requires 3 MVPs with complexity $\mathcal{O}(wk), \mathcal{O}(k^2)$, and $\mathcal{O}(k)$, respectively, and activation computation with complexity $\mathcal{O}(k)$. Given $k > w$ in the DQN, the complexity of the online DQN algorithm is $\mathcal{O}(k^2)$ and it is independent of the size of the power grid.

Table 1
Number and complexity of vector and matrix operations.

Operation	Number	Complexity
VA	7	$\mathcal{O}(M+N)$
MVP	2	$\mathcal{O}(N(M+N))$
MMP	6	$\mathcal{O}(N(M+N)^2)$
MI	2	$\mathcal{O}(N^3)$

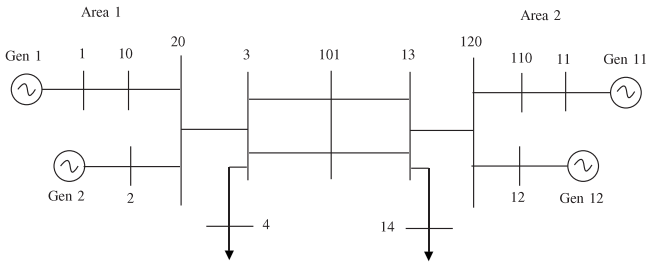

Fig. 2. 13-bus Two Area System [36].

6. Simulation results

6.1. System setup

The simulations are performed on a 13-bus system with two areas as shown in Fig. 2 using MATLAB Power System Toolbox (PST V3.0) [37]. Bus 1 is used as the reference bus. The measurement vector consists of $m = 55$ components, including the voltage magnitude of bus 1, the active and reactive power injections at all 13 buses, and the active and reactive power flows at all 14 lines. The state vector consists of $n = 25$ components, which are the voltage magnitudes at all 13 buses and the phase angles at the 12 non-reference buses. The time interval of the simulation is $\Delta t = 0.01s$, which corresponds to a sampling rate of 100 Hz. The maximum length of each episode is $T = 200$ samples, which corresponds to a time duration of 2 s. The measurement and state vectors are considered as the true values of $\mathbf{z}$ and $\mathbf{x}$, respectively. For the AC system DSE parameters, the covariance matrix of measurement error is set as $\mathbf{R} = \text{diag}(10^{-5}, 10^{-6}, \ldots, 10^{-6})$. The parameters for Holt's linear trend method are $\alpha = 0.95$ and $\beta = 0.001$. The covariance matrix of state transition error is $\mathbf{Q}_t = 10^{-6}\mathbf{I}_n$.

The DQN algorithm is implemented in Python using Stable Baselines [38], with which we built a customized power system environment for our simulations. Hyperparameters are tuned manually based on the episode reward curve and state distributions in Tensorboard. The Q-network and the target network each have two hidden layers with 32 nodes per layer. The discount factor is $\gamma = 1$, and the learning rate for updating θ in (46) is 0.0005. The size of the buffer D is 50,000. The exploration probability of the ϵ-greedy policy is $\epsilon = 0.1$, and the number of transitions in a mini-batch is $|B| = 32$. The frequency of the target network update is $C = 500$.

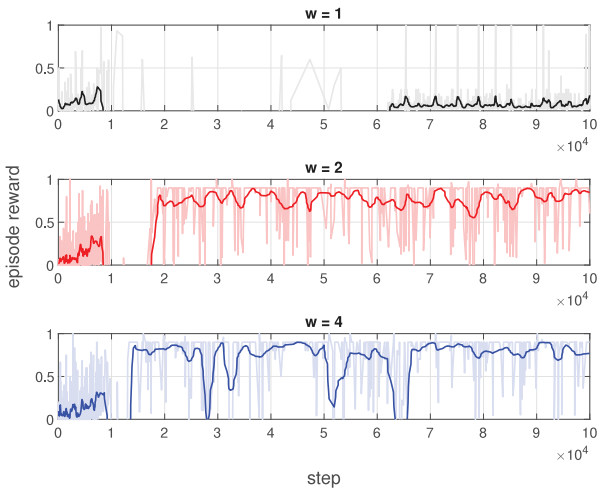

Fig. 3. Learning curve for $\phi = 0.1$ and different w.

6.2. Training results

During the training stage, the number of buses under attack is generated by uniformly sampling from $\{1, 2, \ldots, m\}$. Once the number of buses under attack is determined, the set of indices of buses under attack, $\mathcal{I}$, are sampled uniformly without replacement from the index set of all buses. The attack time τ is uniformly sampled from $\{1, 2, \ldots, T = 200\}$. The elements of the attack vector $\mathbf{a}$ are uniformly sampled from $[-2, 2]$ p.u. for FDI attacks. The system is only trained for FDI attacks. The model trained with FDI attacks will be tested against both FDI and DoS attacks during the testing stage. In addition to cyberattacks, the normal system dynamic and state transitions of the power grid are simulated by increasing the active load at bus 4 by 0.5 per unit (p.u.) at $t = 0$. Model trained under such a deterministic load change will be tested against systems with random load changes during the testing stage. The length of the sliding window used in the MDP state $o(t)$ in (35) is $w \in \{1, 2, 4\}$. The trade-off parameter ϕ used in the MDP reward function in (37) is chosen from $\{0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1\}$. Each agent with a given set of parameters is trained for $E = 10^5$ episodes, which takes about 4 h on a workstation with a 6-core Intel Core i7-5820 K CPU operating at 3.3 GHz and 32 GB of random access memory (RAM).

Fig. 3 shows the episode reward curves for $w \in \{1, 2, 4\}$ and $\phi = 0.1$. The shadow lines are the real episode rewards and the solid lines are the episode rewards after a Gaussian-weighted moving average over 20 consecutive samples. Only the positive episode reward is shown in the figure. The agent with $w = 1$ failed to reach a high episode reward after being trained for $E = 10^5$ episodes, thus it failed to learn a useful detection strategy. This is due to the fact that it makes decisions based on the Rao-test statistic from only the current measurement while ignoring all previous measurements. The training results for agents with $w = 2$ and $w = 4$ successfully reached an episode reward that is close to 1 after being trained for 2×10^4 episodes. The models obtained from the training stage are then used during the testing stage.

6.3. Testing results

The DQN models are tested on the same power system but with different system dynamics and random cyberattacks. For FDI attacks, the attack indices, attack time, and attack vector values are all randomly generated by following the same distributions as described in the training stage. For DoS attacks, the attack indices are generated as the same way as FDI attacks, and the attack matrix $\mathbf{A}$ is generated according to (20). Each agent is tested for 1,000 Monte Carlo simulations, the PFA and ADD are calculated from the simulated detection results according to (32) and (33).

Algorithm 2 DQN Testing: Online Detection

Require: Target Q-network parameters θ' obtained from Algorithm 1, network measurements, episode length T.

Initialization: $t \leftarrow 1$; $\hat{\tau} = \infty$.
while $t \leq T$ **do**
  Calculate $o(t)$.
  Update $a(t)$ as

$$a(t) \leftarrow \arg\max_a Q(o(t), a; \theta')$$

  if $a(t) = 1$ **then**
    $\hat{\tau} \leftarrow t$
    Break
  end if
  $t \leftarrow t + 1$
end while
Ensure: $\hat{\tau}$

Table 2
FDI testing results (PFA, ADD).

Parameters	$\phi = 0.5$	$\phi = 1$
$w = 1$	0.993, 0	0.999, 0
$w = 2$	0.029, 2.0974e−2	0.034, 2.0768e−2
$w = 4$	0.033, 4.0513e−2	0.036, 4.0156e−2

During one testing episode (Monte Carlo trial), the agent works as an online detector. At the tth time-step, it obtains the real measurement $\mathbf{z}_t$ and then calculates the MDP state $o(t)$ from $\mathbf{z}_t$ and historical data. The optimal action is made according to $a(t) = \arg\max_a Q(o, a; \theta')$, where θ' are the target network parameters obtained through the training stage. The testing episode ends if an attack is detected or the end of the episode is reached. Detailed testing procedures in each testing episode are presented in Algorithm 2.

The testing results of FDI attacks for agents with $w \in \{1, 2, 4\}$ and $\phi \in \{0.5, 1\}$ are given in Table 2, where each entry represents the (PFA, ADD) pair obtained for a given configuration. As discussed in Section 5.1, the parameter ϕ can be used to tune the trade-off between ADD and PFA, with a larger ϕ leading to a bigger penalty for detection delay. Such a trade-off relationship can be observed in Table 2, where increasing ϕ from 0.5 to 1 leads to a shorter ADD but a slightly larger PFA. As shown in Fig. 3, the agent with $w = 1$ fails to learn during the training phase, thus the testing PFA is close to 1. The agent with $w = 1$ is just a one-shot soft threshold detector without utilizing historical data. Increasing w from 1 to 2 or 4 leads to systems with considerably better performance. The agents with $w = 2$ slightly outperform their $w = 4$ counterparts in terms of both PFA and ADD. In addition, the model complexity of $w = 2$ is lower than that of $w = 4$ due to the lower dimension of $o(t)$.

The performances of the proposed DQN-based detector under FDI and DoS attacks are shown in Figs. 4 and 5, respectively, where the ADD is plotted as a function of the PFA. In the simulations we have $w = 2$, and the results are compared to that from the Rao-CUSUM detector [33]. The multiple points on the ADD-PFA trade-off curve of the DQN detectors are obtained by setting $\phi \in \{0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1\}$. The trade-off curve of the Normalized Rao-CUSUM detector is obtained by choosing different values for detection threshold as described in [33]. Every point on the curves is obtained by 1,000 Monte Carlo trials. The proposed DQN-based detector outperforms the Rao-CUSUM detector in terms of both PFA and ADD under both FDI and DoS attacks. Note that the ADD for both detectors are based on the time interval of the simulation. In practice, the SCADA updates every 2–5 s, the computation time of DSE in our detector is within 5s for a system

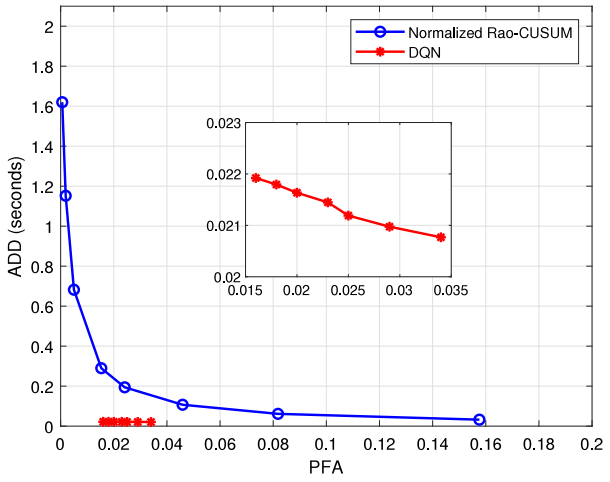

Fig. 4. Performance of DQN detector ($w = 2$) and Normalized Rao-CUSUM detector [33] under FDI attack.

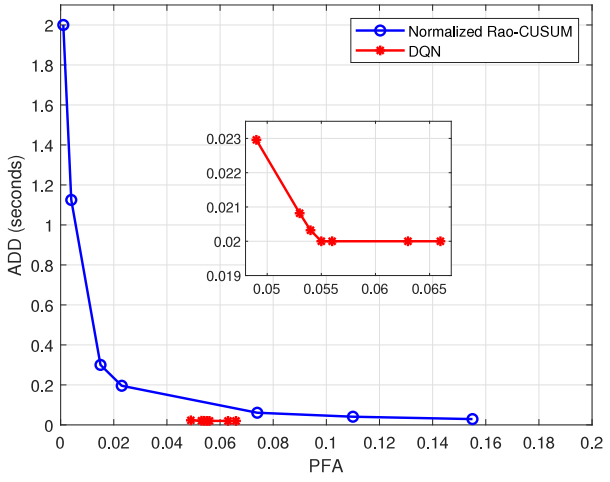

Fig. 5. Performance of DQN detector ($w = 2$) and Normalized Rao-CUSUM detector [33] under DoS attack.

with 200 buses or less [39], and the complexity of online the online DQN detection process is much smaller compared to that of DSE. Since the DQN model is trained under FDI attacks, systems with FDI attacks slightly outperform those with DoS attacks.

Fig. 6 shows the real power at bus 14 under FDI attacks. In case of FDI attacks, the real power measurement at bus 13 is falsely decreased by 1.5 p.u. and that at bus 14 is falsely increased by 1 p.u. between 0.25 and 0.6 s. The proposed DQN-based detector can correctly detect the presence of FDI. Upon detection of FDI, we can remove the false data and replace them with estimated and predicted power values, which are very close to their true values.

7. Conclusion

A DQN-based deep reinforcement learning algorithm has been proposed for the low latency detection of cyberattacks, such as FDI and DoS attacks, in smart grids. Unlike conventional detection methods that focus solely on detection accuracy, the proposed algorithm aims at minimizing the average detection delay while maintaining a low probability of false alarm. The design objective was achieved by developing a DQN-based reinforcement learning algorithm with dynamic AC power system models, which can accurately model dynamic state transients in power systems and identify cyberattacks in real time. The DQN-based reinforcement learning algorithm was developed by following

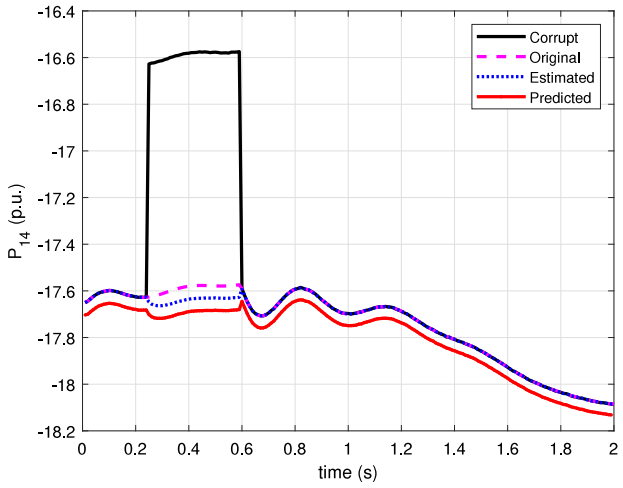

Fig. 6. The real power at bus 14 with FDI at $0.25 < t < 0.6$.

an MDP framework. The MDP state was formulated by using a sliding window of Rao-statistics that can accurately capture the dynamic state evolution of the power grid in real time. A new reward function was designed to allow a flexible trade-off between ADD and PFA. Simulation results demonstrated that the DQN-based RL detection algorithm can achieve very low detection delays while maintaining good PFA performance, and it can achieve considerable performance gains over the existing Rao-CUSUM algorithm. For future works, we plan to apply and improve the proposed algorithm to more sophisticated cyberattacks, e.g., cyberattacks generated by using machine learning algorithms such as generative adversary network (GAN).

CRediT authorship contribution statement

Yaze Li: Methodology, Software, Validation, Investigation, Formal analysis, Writing – original draft, Visualization. **Jingxian Wu:** Conceptualization, Resources, Data curation, Writing – review & editing, Supervision, Project administration, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Greer C, Wollman DA, Prochaska DE, Boynton PA, Mazer JA, Nguyen CT, et al. Nist framework and roadmap for smart grid interoperability standards, release 3.0. Tech.rep., 2014.
[2] Case DU. Analysis of the cyber attack on the ukrainian power grid. In: Electricity information sharing and analysis center. Vol. 388. 2016.
[3] Musleh AS, Chen G, Dong ZY. A survey on the detection algorithms for false data injection attacks in smart grids. IEEE Trans Smart Grid 2019;11(3):2218–34.
[4] Liu Y, Ning P, Reiter MK. False data injection attacks against state estimation in electric power grids. ACM Trans Inf Syst Secur 2011;14(1):1–33.
[5] Huseinovic A, Mrdovic S, Bicakci K, Uludag S. A survey of denial-of-service attacks and solutions in the smart grid. IEEE Access 2020.
[6] Monticelli A. Electric power system state estimation. Proc IEEE 2000;88(2):262–82.
[7] Moslemi R, Mesbahi A, Velni JM. A fast, decentralized covariance selection-based approach to detect cyber attacks in smart grids. IEEE Trans Smart Grid 2017;9(5):4930–41.
[8] Akingeneye I, Wu J. Low latency detection of sparse false data injections in smart grids. IEEE Access 2018;6:58564–73.
[9] Zhao J, Gómez-Expósito A, Netto M, Mili L, Abur A, Terzija V, et al. Power system dynamic state estimation: Motivations, definitions, methodologies, and future work. IEEE Trans Power Syst 2019;34(4):3188–98.

[10] Khalaf M, Youssef A, El-Saadany E. Detection of false data injection in automatic generation control systems using kalman filter. In: 2017 IEEE electrical power and energy conference. IEEE; 2017, p. 1–6.
[11] Kurt MN, Yılmaz Y, Wang X. Distributed quickest detection of cyber-attacks in smart grid. IEEE Trans Inf Forensics Secur 2018;13(8):2015–30.
[12] Karimipour H, Dinavahi V. On false data injection attack against dynamic state estimation on smart power grids. In: 2017 IEEE international conference on smart energy grid engineering. IEEE; 2017, p. 388–93.
[13] Karimipour H, Dinavahi V. Robust massively parallel dynamic state estimation of power systems against cyber-attack. IEEE Access 2017;6:2984–95.
[14] Li Y, Li Z, Chen L. Dynamic state estimation of generators under cyber attacks. IEEE Access 2019;7:125253–67.
[15] Hasnat MA, Rahnamay-Naeini M. A data-driven dynamic state estimation for smart grids under DoS attack using state correlations. In: 2019 North American power symposium. IEEE; 2019, p. 1–6.
[16] Ding Y, Liu J. Real-time false data injection attack detection in energy internet using online robust principal component analysis. In: 2017 IEEE conference on energy internet and energy system integration. IEEE; 2017, p. 1–6.
[17] He Y, Mendis GJ, Wei J. Real-time detection of false data injection attacks in smart grid: A deep learning-based intelligent mechanism. IEEE Trans Smart Grid 2017;8(5):2505–16.
[18] Wei L, Gao D, Luo C. False data injection attacks detection with deep belief networks in smart grid. In: 2018 Chinese automation congress. IEEE; 2018, p. 2621–5.
[19] Fenza G, Gallo M, Loia V. Drift-aware methodology for anomaly detection in smart grid. IEEE Access 2019;7:9645–57.
[20] Ashrafuzzaman M, Chakhchoukh Y, Jillepalli AA, Tosic PT, de Leon DC, Sheldon FT, et al. Detecting stealthy false data injection attacks in power grids using deep learning. In: 2018 14th International wireless communications & mobile computing conference. IEEE; 2018, p. 219–25.
[21] Jaeger H. The "echo state" approach to analysing and training recurrent neural networks-with an erratum note. In: Bonn, Germany: German national research center for information technology GMD technical report. Vol. 148. (34):Bonn; 2001, p. 13.
[22] Hamedani K, Liu L, Atat R, Wu J, Yi Y. Reservoir computing meets smart grids: Attack detection using delayed feedback networks. IEEE Trans Ind Inf 2017;14(2):734–43.
[23] Hamedani K, Liu L, Hu S, Ashdown J, Wu J, Yi Y. Detecting dynamic attacks in smart grids using reservoir computing: A spiking delayed feedback reservoir based approach. IEEE Trans Emerg Top Comput Intell 2019;4(3):253–64.
[24] James J, Hou Y, Li VO. Online false data injection attack detection with wavelet transform and deep neural networks. IEEE Trans Ind Inf 2018;14(7):3271–80.
[25] Kurt MN, Ogundijo O, Li C, Wang X. Online cyber-attack detection in smart grid: A reinforcement learning approach. IEEE Trans Smart Grid 2018;10(5):5174–85.
[26] Chen Y, Huang S, Liu F, Wang Z, Sun X. Evaluation of reinforcement learning-based false data injection attack to automatic voltage control. IEEE Trans Smart Grid 2018;10(2):2158–69.
[27] An D, Yang Q, Liu W, Zhang Y. Defending against data integrity attacks in smart grid: A deep reinforcement learning-based approach. IEEE Access 2019;7:110835–45.
[28] Page ES. Continuous inspection schemes. Biometrika 1954;41(1/2):100–15.
[29] Gardner Jr ES, McKenzie E. Forecasting trends in time series. Manage Sci 1985;31(10):1237–46.
[30] Da Silva AL, Do Coutto Filho M, De Queiroz J. State forecasting in electric power systems. In: IEE proceedings C (generation, transmission and distribution). Vol. 130. (5):IET; 1983, p. 237–44.
[31] Nishiya K, Hasegawa J, Koike T. Dynamic state estimation including anomaly detection and identification for power systems. In: IEE proceedings C (generation, transmission and distribution). Vol. 129. (5):IET; 1982, p. 192–8.
[32] De Maio A. Rao test for adaptive detection in Gaussian interference with unknown covariance matrix. IEEE Trans Signal Process 2007;55(7):3577–84.
[33] Nath S, Akingeneye I, Wu J, Han Z. Quickest detection of false data injection attacks in smart grid with dynamic models. IEEE J Emerg Sel Top Power Electr 2019.
[34] Lovejoy WS. A survey of algorithmic methods for partially observed Markov decision processes. Ann Oper Res 1991;28(1):47–65.
[35] Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, et al. Human-level control through deep reinforcement learning. Nature 2015;518(7540):529–33.
[36] Rogers G. Power system oscillations. Springer Science & Business Media; 2012.
[37] Chow JH, Cheung KW. A toolbox for power system dynamics and control engineering education and research. IEEE Trans Power Syst 1992;7(4):1559–64.
[38] Hill A, Raffin A, Ernestus M, Gleave A, Kanervisto A, Traore R, et al. Stable baselines. 2018, GitHub Repository GitHub. https://github.com/hill-a/stable-baselines.
[39] Karimipour H, Dinavahi V. Extended Kalman filter-based parallel dynamic state estimation. IEEE Trans Smart Grid 2015;6(3):1539–49.