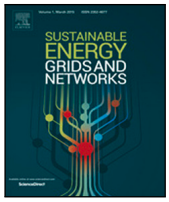
Sustainable Energy, Grids and Networks 38 (2024) 101374

Contents lists available at ScienceDirect

Sustainable Energy, Grids and Networks

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



Multivariate time-series cyberattack detection in the distributed secondary control of AC microgrids with convolutional neural network autoencoder ensemble

Behshad Roshanzadeh *, Jeewon Choi, Ali Bidram, Manel Martínez-Ramón

Department of Electrical and Computer Engineering, The University of New Mexico, NM, USA

ARTICLE INFO

Keywords:
Convolutional neural networks
Cyberattack detection
Deep learning
Microgrid control

ABSTRACT

This paper proposes an unsupervised machine learning-based approach for cyberattack detection in AC microgrids with distributed secondary control architecture. The proposed approach is fully unsupervised and only utilizes the system's normal datasets for the training of the algorithm. The attack under study is a false data injection (FDI) attack tampering with the operating frequency of inverter-based distributed generators (DGs). The paper utilizes a 1D Convolutional Autoencoder (CAE) for cyberattack detection on a microgrid's distributed secondary frequency control. An autoencoder is a neural network architecture, where the model is trained to reconstruct its input in an unsupervised manner. CAE can be applied to a time-series dataset to extract features and exploit the known correlation between neighboring temporal features. Due to the correlation between the operating frequency of DGs and their active power ratios, the paper uses the time series of these two variables as inputs to CAE. The effectiveness of the proposed approach has been verified using a simulated microgrid test system in Matlab/Simulink.

1. Introduction

Due to the vast utilization of advanced communication, control, and monitoring technologies modern electric power grids are prone to cyberattacks at different levels [1]. These technologies have added a new layer to the conventional physical layer of power grids, namely the cyber layer. Cyber threats can not only target the communication system of the cyber layer but also the intelligent electronic devices (IED) used for control, protection, and monitoring applications. According to [2], there are around sixty-three different types of cyberattacks that can potentially target electric grids at different levels. Among those, False data injection (FDI) attacks are common types of attacks that launch control and monitoring units, tamper with the data transferred through the communication links, and affect its integrity [3–5]. On the other hand, a Denial-of-Service (DoS) attack is another common type of attack that can limit the availability of communication networks for data transfer among IEDs. Cyberattacks can have detrimental impacts on power grids such as cascading failures and power outages for customers as well as blackouts by (i) making the system unstable from the voltage and frequency perspective, (ii) deteriorating the performance of power system controllers, and (iii) exposing lines and transformers to overloading conditions by violating their thermal limits. Cyberattacks can also be initiated by attackers to mask a physical attack may lead to cascading failures and long-term power outages for customers. Cyberattacks are identified as threatening risks for all countries in the next ten years according to the World Economic Forum [6]. In [2], some of the real cyberattacks are surveyed which are the Slammer worm of the David-Besse nuclear plant in Ohio in 2003, the Ukraine cyberattack in 2015 with 225,000 customers affected, Malware Triton in the Saudi Arabian oil refinery in 2017, a cyberattack in electric power utilities of US in 2019, and India's Kudankulam Nuclear Power Plant attack in 2019.

The focus of this paper is on the cyber security of microgrids. Microgrids facilitate the integration of renewable energy resources and can provide a reliable source and delivery of power to their customers since they can operate in both grid-connected and islanded modes. To accommodate a reliable operation in both operating modes, microgrids are equipped with a hierarchical control structure consisting of primary, secondary, and tertiary control levels [7,8]. Among these control levels, secondary and tertiary control levels highly rely on a communication network to operate. The secondary control level can adopt both traditional centralized and advanced distributed communication architectures to perform frequency restoration and voltage regulation [7,9–12]. In this paper, we focus on the distributed secondary control of microgrids since an FDI attack on one of the distributed generators

* Corresponding author.
E-mail address: broshan@unm.edu (B. Roshanzadeh).

https://doi.org/10.1016/j.segan.2024.101374
Received 20 February 2023; Received in revised form 9 March 2024; Accepted 2 April 2024
Available online 4 April 2024

(DGs) can be easily propagated to other DGs through the distributed communication network.

Cyberattack detection in microgrids has been addressed in the literature extensively. FDI attacks have been introduced as common types of attacks in microgrids and most of the existing techniques utilize data-driven approaches for this purpose. For centralized communication architectures, Kalman filters [13], adaptive cumulative sum using Markov-chain analysis [14], matrix separation technique, graphical method [15], sparse optimization [16], and cosine similarity matching and Chi-square detector [17]. In [18], an anomaly detection technique, centered around prediction intervals, is presented to differentiate between malicious attacks of varying severity levels during a secure operation. In [19], deep learning along with the Wavelet Singular Values approach are utilized for FDI attack detection in DC microgrids. The cyberattack detection for the distributed secondary control of microgrids has been addressed in [4,5,20]. Signal temporal logic is used for cyberattack detection of a distributed control of a microgrid in [20]. Cyberattack detection using the Kullback–Leibler (KL) divergence for both AC and DC microgrids is addressed in [4,5]. An observer-based approach for attack detection in distributed control of microgrids is provided in [21]. Supervised artificial neural networks and deep learning algorithms are used in [22,23] for cyberattack detection in power grids. An artificial neural network is also used in [24] for attack detection in microgrids. In [25], reinforcement learning is utilized to generate data required for training artificial neural networks for attack detection in DC microgrids. The literature review shows that the existing cyberattack detection techniques (like supervised machine learning algorithms) need a large amount of data under attack to train. This is hard to gather since cyberattacks have low probability and occurrence in power grids. Unsupervised cyberattack detection has been addressed in [26,27]. In [26], feature extraction with symbolic dynamic filtering (SDF) is utilized for feature extraction for anomaly detection in a smart grid. This approach is based on the statistical correlation between measurements received from sensors in a conventional power grid. Ref. [27] utilizes Long Short-Term Memory (LSTM) Stacked Autoencoder (SAE) for attack detection in DC microgrids. While LSTM SAE can provide high accuracy in attack detection, its implementation requires extensive memory and is not computationally efficient compared to convolutional neural networks (CNN).

This paper proposes an approach for cyberattack detection of AC microgrids in the presence of a distributed secondary control. The proposed approach is based on unsupervised CNN for detecting FDI attacks. The considered FDI attack changes the operating frequency of inverter-based DGs in an islanded microgrid. Due to the extensive variety of cyberattacks and the difficulty of gathering sufficient training data for supervised machine learning algorithms, it is of particular importance to adopt unsupervised machine learning algorithms that can detect previously unseen cyberattacks. Moreover, the unsupervised algorithms should require a low number of nontrainable parameters that can be chosen by inference from the available data or by reasonable heuristics. This paper utilizes a 1D Convolutional Autoencoder (CAE) for cyberattack detection on microgrids distributed secondary frequency control. An autoencoder is a neural network architecture, where the model is trained to reconstruct its input in an unsupervised manner. CAE can be applied to a time-series dataset to extract features and exploit the known correlation between neighboring temporal features. The advantage of 1D CNN to its peers like Recurrent Neural Networks (RNN) and Temporal Convolutional Networks (TCN) are: (i) 1D CNNs can process input sequences in parallel, making them more computationally efficient. (ii) 1D CNNs are effective in capturing global dependencies in data, as they consider entire input windows simultaneously. (iii) 1D CNNs have inherent regularization benefits through weight sharing and pooling layers, which can help prevent overfitting. (iv) 1D CNNs are very effective and efficient tools for feature extraction. (v) 1D CNNs are easier to implement than TCN. (vi) 1D CNNs are more computationally efficient and memory-efficient compared to TCNs which makes them a good candidate for hardware implementation. Without loss of generality, it is assumed that FDI attack tampers with the operating frequency of DGs. Due to the correlation between the operating frequency of DGs and their active power ratios, we use the time series of these two variables as inputs to CAE. CAE only needs to be trained with system normal data (i.e., no data under attack is required). The paper has verified the effectiveness of the proposed approach using a simulated microgrid test system in Matlab/Simulink.

This paper is an extension of the previous work of authors in [28]. In the previous work of authors in [28], a linear (dot product) kernel Gaussian Process Regression (GPR) was used to estimate the frequency of DGs using instant values of the active power ratios as the inputs. GPR was trained using the normal data only, i.e., data that did not contain anomalies. The strategy adopted in the GPR is to learn the relationship between the active power ratio (input or predictor) and the frequency (used as desired output or regressor). The GPR measures the likelihood of the predicted frequency error and a one-class SVM applied to the output of the GP raises a flag if this likelihood is low enough to be classified as an anomaly. This setup makes the approach unsupervised, as the training does not require labeled data. In the current paper, we have replaced the GP with a convolutional neural network (CNN). The advantages of using CNN compared to the Gaussian Process are as follows:

1. Instead of using instantaneous values of power ratios as inputs, the 1D CNN has more flexibility to use an input corresponding to a sliding window of the observed time series of both the frequency and active power ratios as the inputs. The 1D Convolutional Autoencoder (CAE) learns the features of the time series of frequency and active power ratio. The 1D CAE can technically identify anomalies on any of its inputs.
2. CNN has a more efficient performance on large datasets which makes it more suitable for large-scale microgrids. While the GPR needs a block training (training with all the available training data at the same time) with a computational burden of $O(N^3)$, N being the number of training samples, the CNN is trained in mini-batches, which eases the computational burden and allows to incrementally train the structure as new data is available. This, when the structure of the CNN has a moderate number of parameters compared to the number of data (as in the case of our approach), allows the CNN to extract non-stationary patterns in the data and changes of relationships over time (in other words, it adapts to the nonstationary environment) which is more suitable for dynamic environments like microgrids.
3. CNN can automatically learn hierarchical features from the data which helps with capturing complex patterns and representations in the input data. However, the effectiveness of the GP heavily depends on the proper selection of a kernel function to learn explicit features.

The contributions of this paper are as follows:

- An unsupervised cyberattack detection scheme for the distributed secondary control of microgrids is proposed.
- The proposed approach only requires system normal simulation data to be trained.
- The proposed approach utilizes CAE as an unsupervised CNN algorithm with a limited number of parameters to be tuned.

The rest of the paper is organized as follows. Section 2 discusses the microgrid's primary and secondary control levels. In Section 3, first, the FDI attack under study is introduced. Then, the details of the cyberattack detection algorithm using CAE are proposed. Section 4 uses simulated data from a microgrid test system to verify the effectiveness and accuracy of the proposed attack detection scheme. Section 5 concludes the paper.

2. Microgrid hierarchical control

This paper addresses the cyberattack detection of an islanded AC microgrid comprised of inverter-based DGs. A microgrid is controlled by its hierarchical control system which consists of three control levels, i.e., primary, secondary, and tertiary controls [7,8]. The primary control is implemented through the so-called droop techniques to provide frequency and voltage stability in the microgrid after islanding. On the other hand, the secondary control level's goal is to restore the microgrid's frequency or regulate its voltage. Herein, the impact of cyberattacks on the microgrid's primary and secondary frequency control is of concern. The relationships between the distributed secondary control, primary control, and DG's internal control loops are illustrated in Fig. 1.

In the microgrid's hierarchical control, the primary control's frequency droop technique is implemented at each DG locally. The frequency droop in ith DG is formulated as

$$\omega_i = \omega_{ni} - m_{Pi} P_i \tag{1}$$

where ω_i is the operating angular frequency in ith DG; ω_{ni} is the frequency droop reference value; m_{Pi} is the droop coefficient; P_i is ith DG's active power. The droop control coefficients should be selected based on the maximum available active power at each DG, i.e.,

$$m_{P1} P_{\max,i} = m_{Pj} P_{\max,j}, \forall i, j \tag{2}$$

Not only controlling the microgrid's frequency, but the primary control also ensures that each DG contributes active power based on its maximum available active power. This means that the primary control level satisfies

$$\frac{P_i}{P_{\max,i}} = \frac{P_j}{P_{\max,j}}, \forall i, j \tag{3}$$

where $P_{\max,i}$ is the maximum available active power at ith DG. Equivalently, (2) can be written as

$$m_{P1} P_i = m_{Pj} P_j, \forall i, j \tag{4}$$

As seen in (1), because of the negative slope of $-m_{Pi}$, the ith DG operating frequency, ω_i, will be slightly smaller than ω_{ni} which is initially set based on the nominal frequency of microgrid. This deviation in the microgrid's frequency can be compensated by adjusting ω_{ni}, which is the role of the microgrid's secondary frequency control level.

This paper assumes a distributed secondary frequency control. In the distributed secondary control, distributed control units are locally available on each DG and they can communicate with each other by a communication network in a distributed fashion. The distributed control protocols at each DG are created to adjust ω_{ni} of the frequency droop control. These protocols can be obtained by differentiating the droop characteristic in (1), i.e.,

$$\dot{\omega}_{ni}(t) = \dot{\omega}_i(t) + m_{Pi} \dot{P}_i(t). \tag{5}$$

The auxiliary frequency and active power control variables $u_{\omega i}$ and u_{Pi} are defined as

$$\dot{\omega}_i(t) = u_{\omega i}(t) \tag{6}$$

$$m_{Pi} \dot{P}_i(t) = u_{Pi}(t). \tag{7}$$

The droop reference ω_{ni} is calculated from the auxiliary control inputs using

$$\omega_{ni} = \int (u_{\omega i} + u_{Pi}) dt. \tag{8}$$

The objectives of the secondary frequency control are twofold. The first objective is to restore the microgrid frequency to the reference frequency ω_{ref} which means that all the operating frequencies of DGs should satisfy

$$\omega_1 = \cdots = \omega_N = \omega_{\text{ref}}. \tag{9}$$

The second objective is to ensure that the active powers of DGs satisfy (3). To reach these objectives, the auxiliary frequency and active power control inputs, i.e., (6) and (7), in a distributed manner, are defined as

$$u_{\omega i}(t) = c_{\omega i} \Big(\sum_{j \in \mathcal{N}_i} a_{ij} (\omega_j(t) - \omega_i(t)) + g_i (\omega_{\text{ref}} - \omega_i(t)) \Big) \tag{10}$$

$$u_{Pi}(t) = c_{Pi} \sum_{j \in \mathcal{N}_i} a_{ij} (m_{Pj} P_j(t) - m_{Pi} P_i(t)) \tag{11}$$

where $c_{\omega i}$ and c_{Pi} are two control parameters. a_{ij} denotes the existence of a communication link between ith and jth DGs. If ith DG can receive information from jth DG, then $a_{ij} = 1$, otherwise $a_{ij} = 0$. $\mathcal{N}_i$ denotes the neighbors of ith DG and includes all the DGs that send their information to ith DG. The pinning gain g_i is an indication of whether the node i has access to the reference frequency ω_{ref} or not.

3. Attack detection methodology

This section first describes the FDI attacks in a microgrid control system and then elaborates on the proposed attack detection scheme.

3.1. Description of FDI attack

We assume that the FDI attack targets the frequency control of a microgrid. The primary and secondary distributed control protocols of an inverter-based DG, discussed in Section 2, are implemented on a microprocessor that has communication ports to either be utilized by the microgrid operator for adjusting DG's control parameters or by the distributed secondary control to communicate with neighboring DGs as required in (10) and (11). A cyber attacker can gain access to these communication ports and tamper with control protocols and parameters as discussed in [3]. We assumed that the cyber attacker targets the primary frequency droop in (1) and corrupts its output ω_i with ω_i^a which is the injected false data by the cyber attacker. As illustrated in Fig. 2, in the control system that is operating in a distributed manner, an attack on one DG not only impacts that DG but also the other neighboring DGs receiving information from the attacked DG.

3.2. Attack detection algorithm

In the proposed attack detection approach, each DG is equipped with a CNN-based attack detection block which uses the locally measured variables at the DG location to detect an attack scenario. The attack detection block along with the internal control loops of an inverter-based DG are shown in Fig. 1. As seen, this block uses $m_{Pi} P_i$ and ω_i of DG as two inputs and makes a decision about the status of the attack on that DG. If an attack is detected, the corresponding DG is shut down and attack status is sent to neighboring DGs to keep them informed for taking mitigative actions as described in Section 3.A.

3.2.1. Introduction to autoencoders

The attack detection block (shown in Fig. 1) requires two inputs (i.e., $m_{Pi} P_i$ and ω_i) to make a decision about the status of the attack. To address anomaly detection in multivariate time series, autoencoder-based methodologies were developed and have shown promising results [29–31]. An autoencoder is a neural network architecture, where the model is trained to reconstruct its input. Assuming that the input data is a multidimensional array $\mathbf{X}$, the network architecture is comprised of two parts: an encoder function that compresses the input sample into a compact, hidden representation $\mathbf{H} = \mathbf{f}(\mathbf{X})$ and a decoder function that reconstructs the input data from the hidden representation $\mathbf{R} = \mathbf{g}(\mathbf{H})$ where the output $\mathbf{R}$ has the same dimensionality as $\mathbf{X}$ [32]. Therefore, the learning process can be simply described by minimizing a loss function $\mathcal{L}(\mathbf{X}, \mathbf{R})$ where the dissimilarity of reconstructed output $\mathbf{R}$ from the input data $\mathbf{X}$ is penalized by the loss function. Mean

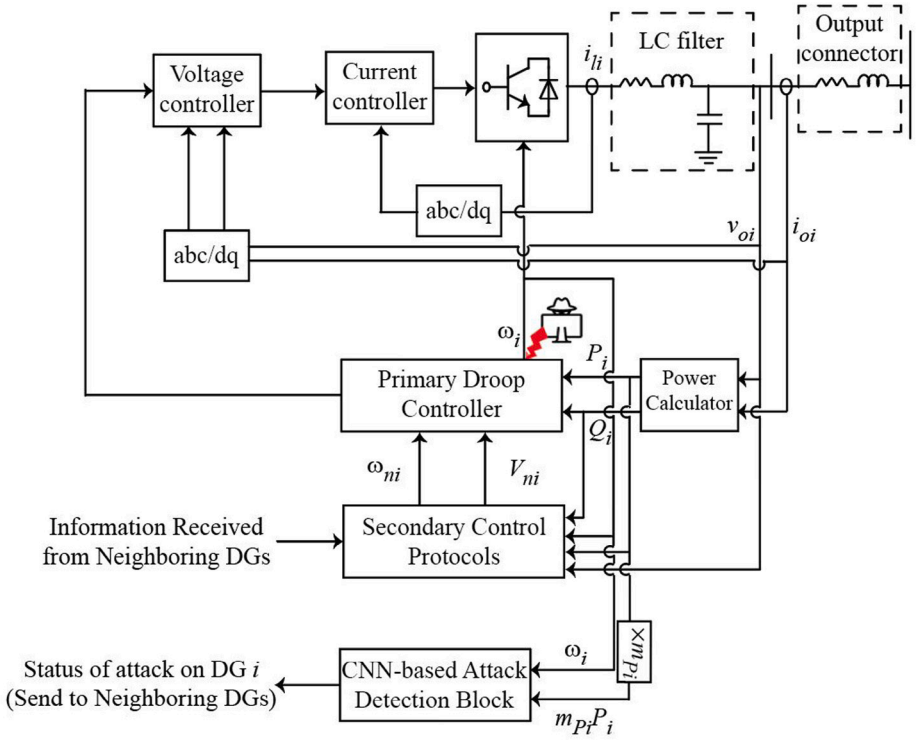


Fig. 1. DG's control diagram including the internal control loops, primary controller, and secondary controller.

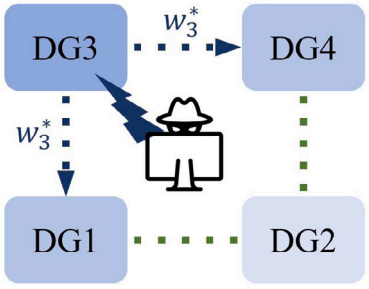


Fig. 2. FDI attack on one DG not only affects the DG under attack but also other DGs that are connected to that DG through the communication network.

absolute error (MAE) and mean squared error (MSE) are two typical choices for loss function. Both of these loss functions are later used in the Ensemble framework discussed in Section 3.B.3. In the case of MAE, the loss function is defined as

$$\mathcal{L}_{MAE}(\mathbf{X},\mathbf{R}) = \frac{\sum_{i=1}^{N} \|\mathbf{X}_i - \mathbf{R}_i\|}{\mathbf{N}} \tag{12}$$

where N is the number of samples we are testing against. In the case of MSE, the loss function is defined as

$$\mathcal{L}_{MSE}(\mathbf{X},\mathbf{R}) = \frac{\sum_{i=1}^{N} \|\mathbf{X}_i - \mathbf{R}_i\|^2}{\mathbf{N}} \tag{13}$$

Since the hidden representation **H** is very compact, only the most representative features of the training data are learned. In this reconstruction-based anomaly detection, the model is trained on normal data with no or very few abnormal samples, so the model learns a reconstruction function that generates a low reconstruction error for normal data and a high reconstruction error for outliers or under attack data. By exploring the range of reconstruction errors/losses for the normal data during training, a threshold can be set and the anomalies can be detected by calculating whether the reconstruction loss is greater than the threshold or not.

3.2.2. CNN-based autoencoder

A 1D convolutional autoencoder (CAE) can be implemented for anomaly detection on a time-series dataset to extract features and for exploiting the known correlation between neighboring temporal features. In this paper, the autoencoder architecture shown in Fig. 3 is utilized as this architecture renders higher accuracy for attack detection. In this architecture, input $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, where $\mathbf{x}_j \in \mathbb{R}^D$ are column vectors, is compressed by the encoder to output a vector **H** of lower dimensionality. The decoder part performs a symmetric reconstruction **R** where the output (reconstructed) layer has the same dimension as the input **X**.

In our data set with two features (i.e., $m_{P_i}P_i$ and ω_i) and a window size of N samples, the input data structure at instant n is

$$\mathbf{X}[n] = \begin{pmatrix} m_{P_i}P_i[n] & \cdots & m_{P_i}P_i[n-N+1] \\ \omega_i[n] & \cdots & \omega_i[n-N+1] \end{pmatrix} \tag{14}$$

The process is started by the convolution of each one of the time series in Eq. (14) by two convolutional filters, each one consisting of a vector of D coefficients. A zero padding is applied to the signals so the output has the same dimension as the input. The output of the convolution is passed through a max pooling, which selects the maximum value of every two consecutive elements of the output of the convolution, resulting in an output of dimension $2 \times N/2$. Each element of the output is applied to a rectified linear unit (ReLU). For an arbitrary input u, the ReLU function is defined as $o = max(0.u)$. After this, a dropout is usually performed in this class of structures, though since the dimension of our present instance is small, this procedure does not improve the performance here. Dropout is a process where a given percentage of connections with the next layer is dropped randomly.

The process is repeated again with convolution kernels of dimension D and another max pooling and ReLu operation, which results in an array of dimension $2 \times N/4$. The output is then processed for the reconstruction of the input. The process starts with a convolution (ConvTranspose in the figure) with a kernel of dimension 5, an upsampling, consisting of repeating each sample, to double the dimension from $D/4$ to $D/2$, and then the process is repeated, to obtain an output of

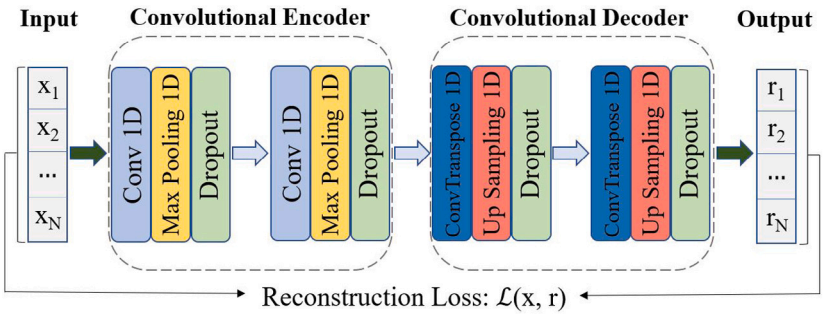


Fig. 3. Architecture of our fully convolutional autoencoder.

dimension $2\times N$. The output is applied to a sigmoid $o=(1+\exp(-u))^{-1}$ instead of a ReLU function, to obtain an output normalized between 0 and 1.

3.2.3. Ensemble framework

Ensembling is another powerful technique that we employed to further improve the accuracy of a single autoencoder which may often overfit to the original data. The key characteristic of an ensemble framework is diversity. Two sources of diversity are introduced into our autoencoder ensemble framework. First, the ensemble models are fit on two different loss functions, i.e., MAE and MSE in (12) and (13), respectively. Second, a bagging procedure [33] is performed by training models with different random initializations. Our ensemble framework consists of 50 models in total. Once an ensemble framework is trained, the reconstruction loss for input data can be calculated by comparing the reconstructed time series to its original input for each model using an MAE loss function. In this work, the median is used as an ensemble aggregation function.

3.2.4. Evaluation metrics

Precision (PR), recall (RE), and F1-score are used over the testing set and its ground truth values to evaluate the performance of our ensemble autoencoder model. F1-score can be formulated as [34]

$$\mathrm{F1}=\frac{2\times \mathrm{PR}\times \mathrm{RE}}{\mathrm{PR}+\mathrm{RE}} \tag{15}$$

where

$$\mathrm{PR}=\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}} \tag{16}$$

$$\mathrm{RE}=\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}} \tag{17}$$

TP, TN, FP, and FN are the values of true positives, true negatives, false positives, and false negatives, respectively. Since these metrics are more resistant to class imbalance than other metrics such as Accuracy [34], they are well-suited for anomaly detection tasks. For each model within the ensemble framework, the threshold is defined as the maximum reconstruction loss. The median is used as an ensemble aggregation function to define the threshold for the whole ensemble.

3.2.5. Attack mitigation strategy

The attack mitigation strategy can be adopted from the previous work of authors in [28]. If an attack is detected by the attack detector of DG i, the associated DG will be disconnected from the microgrid to maintain the microgrid's stability. Simultaneously, the information related to the attacked DG will be segregated from the communication network. This is achieved by setting $a_i^*=0$ in the following equation, where the default value is $a_i^*=1$ before detecting any attacks. This altered information is then communicated to the neighboring controllers, ensuring that incoming compromised data is also isolated in those neighboring controllers.

$$\begin{aligned} u_{\omega i}(t)&=c_{\omega i}\sum_{j\in\mathcal{N}_i}a_{ij}a_i^*a_j^*(\omega_j(t)-\omega_i(t))+g_i(\omega_{\mathrm{ref}}-\omega_i(t))\\ u_{Pi}(t)&=c_{Pi}\sum_{j\in\mathcal{N}_i}a_{ij}a_i^*a_j^*(m_{Pj}P_j(t)-m_{Pi}P_i(t)) \end{aligned} \tag{18}$$

4. Verification of proposed approach

4.1. Description of test system

This paper utilizes the 4 DG microgrid test system shown in Fig. 4 to verify the effectiveness of the proposed attack detection algorithm. The specifications of this microgrid test system are provided in Table 1. This microgrid and its specifications are adopted from [3,10]. For the DG parameters, m_P is the frequency droop coefficient, n_Q is the voltage droop coefficient, R_c is DG's output connector resistance, L_c is DG's output connector inductance, R_f is DG's output filter resistance L_f is the DG's output filter inductance, C_f is the DG's output filter capacitance, K_{PV} and K_{IV} are the internal voltage control proportional and integral gains, and K_{PC} and K_{IC} are the internal current control proportional and integral gains. The training and testing datasets are collected for each DG by simulating different normal and cyberattack scenarios in Matlab/Simulink, and the attack detector is trained for each DG. Due to the unsupervised nature of the proposed CNN-based attack detection algorithm, the training is performed using system normal data (in the absence of cyberattacks). The system's normal scenarios include the simulation of microgrids under islanding, load change, and DG connection/disconnection. The simulations account for a load change in the range of 0 to 20 kW.

Each set of data is sampled at every 100 μs. Four datasets corresponding to the system's normal conditions (without attack) are collected that correspond to the transient scenarios of islanding, load changes, and DG connection/disconnection. A portion of these datasets is used to train the CNN-based algorithm. Six sets of data under random attack applied are also collected. These datasets are used for testing the algorithm along with the remaining portion of datasets collected under system normal conditions. Each system normal dataset contains 180 to 250×10^3 samples.

4.2. Description of attack and its impact on the microgrid frequency control

To simulate an FDI attack on the frequency control of the microgrid test system, the frequency of DG 3 in (1) is manipulated with the corrupted frequencies of 58, 59, 60.5, 61, 61.5, and 62 Hz (seven attack scenarios that were simulated separately). The impact of this attack on the microgrid frequency control is studied in the previous work of authors in [28]. When the FDI attack is applied on a DG (i.e., a DG frequency is manipulated with a corrupted frequency), the corrupted frequency is propagated through the microgrid in two ways. First, the corrupted frequency will impact the microgrid's electric system frequency which results in the microgrid frequency stability. Moreover, the corrupted frequency propagates through the communication network of the distributed secondary control. This will directly impact the performance of secondary control as DGs' distributed secondary control protocols will use the corrupted frequency of attacked DG and fail to restore the microgrid's frequency and properly share active power among them.

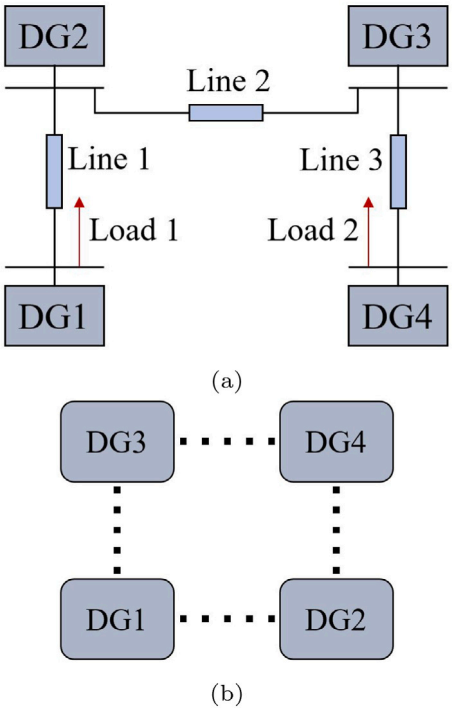


Fig. 4. 4 DG MG test system: (a) circuit diagram; (b) communication graph.

Table 1
Specifications of 4 DG MG test system.

DGs	DG 1, 4	DG 2, 3
m_P	1×10^{-4}	2×10^{-4}
n_Q	2×10^{-3}	4×10^{-3}
R_c	0.05 Ω	0.05 Ω
L_c	4.8 mH	4.8 mH
R_f	0.1 Ω	0.1 Ω
L_f	1.35 mH	1.35 mH
C_f	50 μF	50 μF
K_{PV}	0.1	0.05
K_{IV}	420	390
K_{PC}	15	10.5
K_{IC}	20000	16000

Lines	Line 1	Line 2	Line 3
R	0.2 Ω	0.1 Ω	0.2 Ω
L	3.6 mH	1.8 mH	3.6 mH

Loads	Load 1	Load 2
P_L	12 kW	12 kW
Q_L	5 kVAr	5 kVAr

4.3. Data preparation

For all datasets, some preprocessing is performed to prepare them as inputs to the CNN-based attack detection algorithm. Testing datasets contain both system-normal and under-attack data, whereas training datasets contain only system-normal data. Before feeding the data into the model a few preprocessing steps were taken. First, input features/variables are normalized by transforming them into the range of 0 to 1 so that all the features contribute equally to the model and the learned model is not biased towards one specific feature. The normalization is performed by subtracting the minimum value from the data and then dividing by the difference between the maximum and the minimum. Also, all four normal datasets are concatenated together to form one training set. Similarly, all datasets under random attacks are concatenated to form one testing set.

Our models are exclusively trained on data without attack. After performing the preprocessing, the training dataset is split into 90% training and 10% validating sets. The validation data is used for hyperparameter optimization to minimize the reconstruction loss during the training. The hyperparameters considered in this work are the batch size, the early stopping instant, and the parameters of the Adam optimizer.

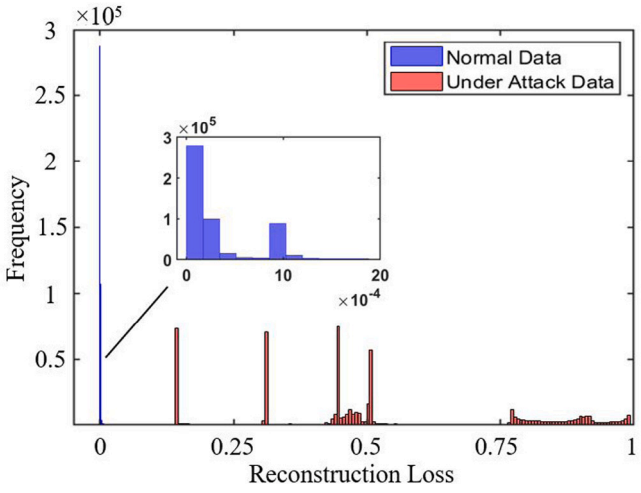


Fig. 5. Distribution of reconstruction loss over the normal and under attack data for DG3 with a window size of 16.

4.4. Implementation and hyperparameters

The framework was implemented with Python and TensorFlow. All presented calculations were executed leveraging Nvidia Tesla T4 GPU on Google Colab.

The neural network training is run with an early stopping criterion by monitoring the loss convergence on the validation set [32]. The batch size is optimized on the validation set as well and has been set to a fixed size of 1024. Stochastic gradient descent enhanced with the Adam optimizer [32] with default settings and an initial learning rate of 0.001 is used. The patience parameter and the reduction factor for the learning rate are set to 10 and 5, respectively. The GPU-based training of one ensemble framework for the entire training set, including all four DGs, takes between 1.5 and 4 h depending on the neural network settings and the window size.

4.5. Verification results

To examine the reconstruction loss in more detail, a histogram of reconstruction error generated by the ensemble model on the normal data from the training set and under-attack data from the test set is visualized in Fig. 5. The attack scenarios include manipulating the frequency of DG 3 with the corrupted frequencies of 58, 59, 60.5, 61, 61.5, and 62 Hz (six attack scenarios that were simulated separately). Each attack scenario has 10^4 normal instances (i.e., before the attack is applied) and 8×10^4 under-attack instances. The input data has a window size of 16. We can see the reconstruction loss is significantly smaller for normal samples leading to a distinct distribution for normal compared to under-attack data. In this figure, for the 58 Hz and 62 Hz attack scenarios, the reconstruction loss distribution is between 0.75 to 1, for the 61 Hz attack scenario, the reconstruction loss distribution is around 0.45, for the 59 Hz attack scenario, the reconstruction loss distribution is around 0.5, for the 60.5 Hz attack scenario, the reconstruction loss distribution is around 0.15, and for the 61.5 Hz attack scenario, the reconstruction loss distribution is between 0.45 to 0.55.

The attack detection accuracy in terms of precision, recall, and F1-score for different window sizes are shown in Table 2. The results are based on aggregate performance metrics of our autoencoder ensemble

Table 2
Attack detection accuracy in terms of F1-score (%), precision (%), and recall (%) for different window sizes on the testing set.

Window size	F1-Score	Precision	Recall
8	99.87	100.0	99.73
16	99.87	100.0	99.73
32	99.86	100.0	99.73

Table 3
Comparing attack detection accuracy in terms of F1-score (%), precision (%), and recall (%) between the CNN autoencoder ensemble approach and the Gaussian Process Regression approach.

Cyberattack detection approach	F1-Score	Precision	Recall
CNN Autoencoder Ensemble	99.97	100.0	99.93
Gaussian Process Regression	99.80	99.9	99.80

framework over the testing set. These results are gathered for the window sizes of 8, 16, and 32. The verification results show that the proposed 1D CAE approach renders very high accuracy even in the presence of an FDI attack where the DG 3 frequency is corrupted with 60.5 Hz which is very close to the nominal frequency of the microgrid.

As shown in Table 2, by increasing the window size in time-series data, the scoring metrics start to deteriorate gradually. Due to the nature of the time series, it is difficult to clearly distinguish between the normal and under-attack sections of the data. Therefore, as the length of window size increases, the probability of a mixed period of normal and under-attack data feeding as input x into the trained model increases, which results in low reconstruction error for the mixed period.

A comparative analysis of cyberattack detection accuracy is shown in Table 3 between the CNN autoencoder ensemble proposed in this study and a previous approach by the authors employing Gaussian Process Regression (GPR) [28]. The attack scenarios in the GPR based cyberattack detection paper include manipulating the frequency of DG 3 with the corrupted frequencies of 58, 59, 61, and 62 Hz. To ensure a direct comparison, identical attack scenarios are utilized as the test dataset for the trained CNN autoencoder model with a window size of 16. CNN based approach shows slightly better detection accuracy compared to GPR based approach. Also, GPR imposes a considerable computational burden compared to CNN autoencoder approach. GPR requires two stages of training involving approximately 0.5 million samples, which incurs a substantial computational load. This is due to the necessity of matrix inversion for GPR and solving a quadratic problem for the Support Vector Machine (SVM) component. Both processes exhibit a computational complexity of $O(N^3)$. Additionally, GPR necessitates block training, requiring periodic retraining based on a secondary criterion. In contrast, CNN can undergo continuous training from the initial batch, enabling its usability and potentially offering adaptive properties.

5. Conclusions and future work

In this paper, we investigated the potential of an autoencoder ensemble framework based on 1D convolutional neural networks for unsupervised FDI attack detection in the distributed secondary frequency control of AC microgrids. The ensemble framework trains multiple autoencoders independently on multivariate time series with different window sizes. Experimental studies show that the proposed autoencoder ensemble can detect attacks with an F1 score of close to one. The reported results are very encouraging, but further research should focus on analyzing more complex neural network architectures in order to detect more sophisticated attacks.

CRediT authorship contribution statement

Behshad Roshanzadeh: Writing – review & editing, Writing – original draft, Validation, Software, Methodology, Conceptualization. **Jeewon Choi:** Visualization, Validation, Investigation. **Ali Bidram:** Writing – review & editing, Writing – original draft, Supervision, Methodology, Conceptualization. **Manel Martínez-Ramón:** Methodology, Conceptualization, Writing – original draft, Writing – review & editing.

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.

Data availability

Data will be made available on request.

Acknowledgments

This material is based upon work supported by the National Science Foundation, United States under Awards OIA-1757207 and ECCS-2214441. Manel Martínez-Ramón is partially supported by the King Felipe VI Endowed Chair of the University of New Mexico.

References

[1] L. Zhang, L. Li, L. Wihl, M. Kazemtabrizi, S.Q. Ali, J.-N. Paquin, S. Labbé, Cybersecurity study of power system utilizing advanced CPS simulation tools, in: Proc. 2019 PAC World Americas Conf., Raleigh, NC, USA, 2019, pp. 19–22.
[2] R.V. Yohanandhan, R.M. Elavarasan, P. Manoharan, L. Mihet-Popa, Cyber-physical power system (cpps): A review on modeling, simulation, and analysis with cyber security applications, IEEE Access 8 (2020) 151019–151064.
[3] A. Bidram, L. Damodaran, R. Fierro, Cybersecure distributed voltage control of AC microgrids, in: 2019 IEEE/IAS 55th Industrial and Commercial Power Systems Technical Conference, I&CPS, IEEE, 2019, pp. 1–6.
[4] B.P. Poudel, A. Mustafa, A. Bidram, H. Modares, Detection and mitigation of cyber-threats in the DC microgrid distributed control system, Int. J. Electr. Power Energy Syst. 120 (2020) 105968.
[5] A. Mustafa, B. Poudel, A. Bidram, H. Modares, Detection and mitigation of data manipulation attacks in AC microgrids, IEEE Trans. Smart Grid 11 (3) (2019) 2588–2603.
[6] World Economic Forum, The Global Risks Report 2019, fourteenth ed., World Economic Forum Geneva, Switzerland, 2019.
[7] A. Bidram, A. Davoudi, Hierarchical structure of microgrids control system, IEEE Trans. Smart Grid 3 (4) (2012) 1963–1976.
[8] J.M. Guerrero, J.C. Vasquez, J. Matas, L.G. De Vicuña, M. Castilla, Hierarchical control of droop-controlled AC and DC microgrids—A general approach toward standardization, IEEE Trans. Ind. Electron. 58 (1) (2011) 158–172.
[9] N.M. Dehkordi, N. Sadati, M. Hamzeh, Distributed robust finite-time secondary voltage and frequency control of islanded microgrids, IEEE Trans. Power Syst. 32 (5) (2017) 3648–3659.
[10] A. Bidram, A. Davoudi, F.L. Lewis, J.M. Guerrero, Distributed cooperative secondary control of microgrids using feedback linearization, IEEE Trans. Power Syst. 28 (3) (2013) 3462–3470.
[11] A. Bidram, A. Davoudi, F.L. Lewis, Z. Qu, Secondary control of microgrids based on distributed cooperative control of multi-agent systems, IET Gener. Transm. Distrib. 7 (8) (2013) 822–831.
[12] A. Bidram, A. Davoudi, F.L. Lewis, A multiobjective distributed control framework for islanded AC microgrids, IEEE Trans. Ind. Inform. 10 (3) (2014) 1785–1798.
[13] K. Manandhar, X. Cao, F. Hu, Y. Liu, Detection of faults and attacks including false data injection attack in smart grid using Kalman filter, IEEE Trans. Control Netw. Syst. 1 (4) (2014) 370–379.
[14] Y. Huang, J. Tang, Y. Cheng, H. Li, K.A. Campbell, Z. Han, Real-time detection of false data injection in smart grid networks: An adaptive CUSUM method and analysis, IEEE Syst. J. 10 (2) (2014) 532–543.
[15] S. Bi, Y.J. Zhang, Graphical methods for defense against false-data injection attacks on power system state estimation, IEEE Trans. Smart Grid 5 (3) (2014) 1216–1227.
[16] L. Liu, M. Esmalifalak, Q. Ding, V.A. Emesih, Z. Han, Detecting false data injection attacks on power grid by sparse optimization, IEEE Trans. Smart Grid 5 (2) (2014) 612–621.

[17] D.B. Rawat, C. Bajracharya, Detection of false data injection attacks in smart grid communication systems, IEEE Signal Process. Lett. 22 (10) (2015) 1652–1656.
[18] A. Kavousi-Fard, W. Su, T. Jin, A machine-learning-based cyber attack detection model for wireless sensor networks in microgrids, IEEE Trans. Ind. Inform. 17 (1) (2020) 650–658.
[19] M. Dehghani, T. Niknam, M. Ghiasi, N. Bayati, M. Savaghebi, Cyber-attack detection in dc microgrids based on deep machine learning and wavelet singular values approach, Electronics 10 (16) (2021) 1914.
[20] O.A. Beg, T.T. Johnson, A. Davoudi, Detection of false-data injection attacks in cyber-physical DC microgrids, IEEE Trans. Ind. Inform. 13 (5) (2017) 2693–2703.
[21] S. Tan, P. Xie, J.M. Guerrero, J.C. Vasquez, R. Han, Cyberattack detection for converter-based distributed dc microgrids: Observer-based approaches, IEEE Ind. Electron. Mag. 16 (3) (2021) 67–77.
[22] S. Sengan, V. Subramaniyaswamy, V. Indragandhi, P. Velayutham, L. Ravi, Detection of false data cyber-attacks for the assessment of security in smart grid using deep learning, Comput. Electr. Eng. 93 (2021) 107211.
[23] E.M. Ferragut, J. Laska, M.M. Olama, O. Ozmen, Real-time cyber-physical false data attack detection in smart grids using neural networks, in: 2017 Intl. Conf. on Comp. Sci. and Comp. Intelligence, CSCI, IEEE, 2017, pp. 1–6.
[24] M.R. Habibi, S. Sahoo, S. Rivera, T. Dragičević, F. Blaabjerg, Decentralized coordinated cyberattack detection and mitigation strategy in DC microgrids based on artificial neural networks, IEEE J. Emerg. Sel. Top. Power Electron. 9 (4) (2021) 4629–4638.
[25] Y. Wan, T. Dragičević, Data-driven cyber-attack detection of intelligent attacks in islanded dc microgrids, IEEE Trans. Ind. Electron. 70 (4) (2022) 4293–4299.
[26] H. Karimipour, A. Dehghantanha, R.M. Parizi, K.-K.R. Choo, H. Leung, A deep and scalable unsupervised machine learning system for cyber-attack detection in large-scale smart grids, IEEE Access 7 (2019) 80778–80788.
[27] A. Takiddin, S. Rath, M. Ismail, S. Sahoo, Data-driven detection of stealth cyber-attacks in dc microgrids, IEEE Syst. J. 16 (4) (2022) 6097–6106.
[28] J. Choi, B. Roshanzadeh, M. Martínez-Ramón, A. Bidram, An unsupervised cyber-attack detection scheme for AC microgrids using Gaussian process regression and one-class support vector machine anomaly detection, IET Renew. Power Gener. (2023).
[29] T. Kieu, B. Yang, C. Guo, C.S. Jensen, Outlier detection for time series with recurrent autoencoder ensembles, in: IJCAI, 2019, pp. 2725–2732.
[30] C. Yin, S. Zhang, J. Wang, N.N. Xiong, Anomaly detection based on convolutional recurrent autoencoder for IoT time series, IEEE Trans. Syst. Man Cybern. 52 (1) (2020) 112–122.
[31] M. Thill, W. Konen, H. Wang, T. Bäck, Temporal convolutional autoencoder for unsupervised anomaly detection in time series, Appl. Soft Comput. 112 (2021) 107751.
[32] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016, http://www.deeplearningbook.org.
[33] L. Breiman, Bagging predictors, Mach. Learn. 24 (2) (1996) 123–140.
[34] Q. Gu, L. Zhu, Z. Cai, Evaluation measures of the classification performance of imbalanced data sets, in: International Symposium on Intelligence Computation and Applications, Springer, 2009, pp. 461–471.