DSI DATA SCIENCE AND INFORMETRICS

RESEARCH ARTICLE

Time series anomaly detection based on GAN-VAE

Shibo Liao, Chunshan Liu*, Yongxiang Xia, Haicheng Tu

School of Communication Engineering, Hangzhou Dianzi University, Hangzhou, Zhejiang, China

ABSTRACT

Time-series anomaly detection is crucial for identifying unusual patterns that deviate from expected behavior in temporal data, enabling early intervention in diverse fields such as finance, healthcare, and cybersecurity. This paper proposes a novel anomaly detection method based on deep learning models including Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs). The proposed GAN-VAE model combines the strengths of GAN and VAE to effectively capture the distribution of time series data and optimize sequence mapping in latent space, achieving high accuracy in the reconstruction of normal time-series. Anomalies can then be detected by identifying abnormally large reconstruction errors. To enhance the convergence of the GAN-VAE model in training, a sequential training method is proposed that trains the encoder, decoder, and discriminator in an alternating fashion. The effectiveness of the proposed anomaly detection method is verified through real-world time series datasets.

KEYWORDS

Anomaly detection; GAN-VAE; Sequential training

1. Introduction

The proliferation of data, driven by significant advances in data collection, transmission, storage, and computation, signals the advent of the "big data" era. Through in-depth analysis and anomaly detection of these large-scale datasets, potential patterns and valuable information can be uncovered. On the one hand, erroneous information in the data, if not effectively addressed, can negatively impact the final scientific analysis. For instance, in information analytics, unprocessed misinformation may lead to incorrect intelligence conclusions, thereby affecting the accuracy and reliability of decision-making (Liu et al., 2024). On the other hand, anomalies may provide accurate indications of unusual events or behaviors that are of interest. As noted in (De Almeida Parizotto et al., 2020), “while in many fields outliers can simply be discarded as being exceptions, in bibliometrics the extreme values represent the high-end of research performance and therefore deserve special attention.”

* Corresponding Author: chunshan.liu@hdu.edu.cn

Time series anomaly detection, as a key step in data analysis, has the core task of accurately identifying abnormal behavior or events from normal time series data (Zehra et al., 2023). Typically, time series anomalies can be classified into three categories (Cui & Wang, 2017): 1) Point anomalies, which are individual anomalous data points that do not conform to the normal behavior of the time series as a whole (e.g., very large/very small outliers). 2) Contextual anomalies, which are anomalous in a particular context and may not be anomalous in a different context. 3) Collective anomalies, which constitute a collection of anomalous data points, such as change points (Zhu et al., 2020). In this paper, we aim to develop an anomaly detector that is able to detect all three types of anomalies in large-scale time series data, which may consist of diverse time-series with different patterns.

Existing anomaly detection approaches can be classified into two categories, i.e., statistical approaches and deep learning (DL) based approaches. Statistical approaches for anomaly detection encompass a variety of underlying principles. For instance, the K-Nearest Neighbor (KNN) approach to anomaly detection assigns anomaly scores by leveraging the distances to the nearest k neighbors' samples (Liu et al., 2017). Conversely, the Autoregressive Moving Average (ARIMA) technique models temporal data trends and evaluates discrepancies between forecasted and actual observations (Tron et al., 2018). The Local Outlier Factor (LOF) was adopted by Xu et al. (2021), and identifies anomalies through analyzing the discrepancy in local density around a data point in comparison to its neighboring points. Notwithstanding, the efficacy of the KNN methodology is significantly influenced by the choice of k-value, and both the ARIMA and LOF frameworks are noted for their substantial demands on computational resources and processing time, particularly in the context of voluminous datasets. This computational intensity becomes a pronounced challenge in the realm of large-scale network traffic data, where these statistical techniques may not suffice in fulfilling the industrial benchmarks for anomaly detection precision (Niu et al., 2020).

DL based anomaly detection approaches seek for DL models to learn the inherent patterns of time-series data, from which a violation of the learned pattern can be declared as an anomaly. In supervised learning tasks that rely on labeled data, anomaly detection can achieve high accuracy if sufficient and accurate labels are provided. However, labeling data can be a challenging and resource-intensive task. In the meanwhile, supervised-learning models often have limited generalizing capability to new and unseen anomaly data, which is not uncommon (Chen et al., 2023).

For this reason, DL models with unsupervised learning have received much attention. For instance, Qin et al. (2018) used LSTM trained to predict future moments and detect anomalies from the residuals between predicted and actual values. Zavrak and Iskefiyeli (2020) used a Variational Autoencoder (VAE) for anomaly detection, using the reconstruction error as the anomaly score, and any data with a score greater than a threshold is considered anomalous. Geiger et al. (2020) leveraged a Generative Adversarial Network (GAN) to compute reconstruction errors using both point-wise and window-based methods, deriving anomaly scores by combining the outputs of the generator and discriminator for anomaly detection. Zhu et al. (2019) utilised the deep feature extraction capability of the LSTM-GAN network by performing feature extraction on the detection data and then sending it to the generator for reconstruction, and the reconstruction error of the reconstructed and detected data with the output of the discriminator is used as the criterion for anomaly detection. Xu et al. (2024) proposed a calibrated one-class classification method to enhance the robustness and accuracy of unsupervised time series anomaly detection through uncertainty modeling and native anomaly calibration. Wang et al. (2024) integrated the frequency feature of time series into VAE to increase the accuracy of the reconstruction of normal data, facilitating more accurate anomaly detections.

As discussed above, many different types of anomaly detection techniques have been developed to tackle the challenges of time series anomaly detection. Of these techniques, conventional statistical methods tend to not scale well with the amount of data. Supervised learning-based approaches face the difficulties of lacking anomaly labels. Unsupervised learning-based approach reflects a

promising trend for large-scale time series anomaly detection. In this work, we propose a new GAN-VAE model based on gated recurrent units (GRUs) for time-series anomaly detection. The proposed model consists of three parts: encoder, decoder and discriminator. In this model, the core idea is to first utilize the VAE network for data reconstruction, and then adopt the reconstruction error as a pivotal criterion for anomaly detection. The GAN network adaptively captures the probability distribution of normal data within the latent space and can enhance the VAE network's capability of time-series reconstruction. Furthermore, a sequential training procedure is employed for the encoder, decoder, and discriminator components of the GAN-VAE architecture. This strategy is pivotal in fine-tuning each segment of the GAN-VAE framework, consequently enhancing the efficiency of the model's convergence.

2. Preliminaries of time-series anomaly detection

For unsupervised DL-based time series anomaly detection, a common approach is to train a DL model to predict or reconstruct the time-series of interest. By comparing the predicted/reconstructed values with the true values, anomalies can be declared when the perdition/reconstruction errors become exceptionally large. In this procedure, the DL model used to model the pattern of the data plays a central role. In this section, we first briefly introduce the DL models commonly used for time-series anomaly detection as well as the key DL models for our proposed approach, and then discuss the operational procedures of the classical algorithms for anomaly detection.

2.1. Model fundamentals

2.1.1. Long Short-Term Memory (LSTM) Model

Long Short Term Memory Network (LSTM), a variant of the recurrent neural network, contains three gates, i.e., forgetting gate, input gate and output gate. The information flow of LSTM can be represented as follows:

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \quad (1.a)$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (1.b)$$

$$\tilde{C}_t = \tan\mathrm{h}(W_C \cdot [h_{t-1}, x_t] + b_C) \quad (1.c)$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \quad (1.d)$$

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \quad (1.e)$$

$$h_t = o_t * \tanh(C_t) \quad (1.f)$$

where σ denotes the logistic sigmoid function, x_t denotes the input vector at the current moment, h_t is the hidden state, W is the weight matrix (e.g., W_i denotes the input gate weight matrix) and b is the bias vector (e.g., b_i denotes the input gate bias vector). Figure 1 shows the structure of the LSTM cell.

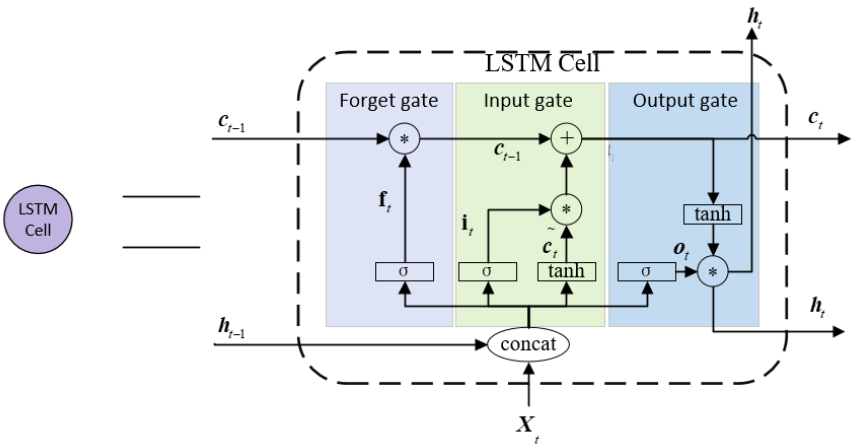

Figure 1 Structure of the LSTM cell.

2.1.2. Gated Recurrent Unit (GRU) Module

GRU is a simplified variant of LSTM which has a similar capability of modelling time-dependent data but with lower computational complexity than LSTM. The information flow of the GRU network can be represented as follows:

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t] + b_r) \quad (2.a)$$

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z) \quad (2.b)$$

$$\tilde{h}_t = \tanh(W_h \cdot [r_t * h_{t-1}, x_t] + b_h) \quad (2.c)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t \quad (2.d)$$

where σ is the logistic sigmoid function, x_t is the current input vector, h_t is the hidden layer, W is the weight matrix, and b is the bias vector. Figure 2 shows the structure of the GRU unit.

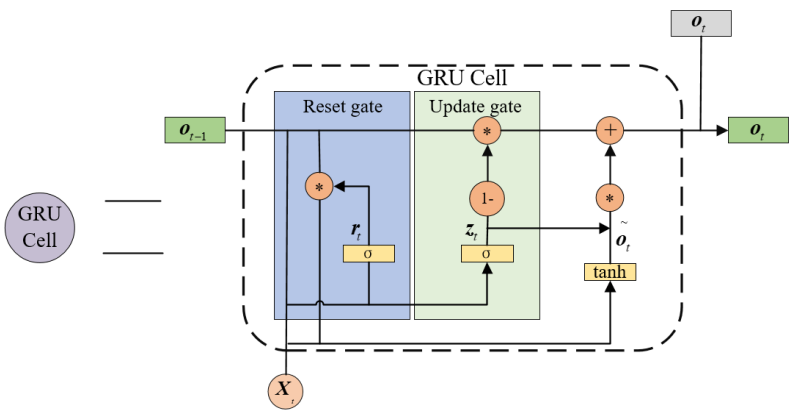

Figure 2 Structure of the GRU cell.

2.1.3. Variational Autoencoder (VAE) Module

VAE is an unsupervised learning model that learns to map input data to a latent space and reconstructs it back to the original data. This process involves an encoder that transforms the input $\boldsymbol{x}$ into a latent variable $\boldsymbol{z}$, and a decoder that reconstructs $\boldsymbol{x}'$ from $\boldsymbol{z}$:

$$\boldsymbol{z} \sim Enc(\boldsymbol{x}) = q(\boldsymbol{z}|\boldsymbol{x}), \qquad \boldsymbol{x}' \sim Dec(\boldsymbol{x}) = p(\boldsymbol{x}|\boldsymbol{z}) \quad (3)$$

The VAE minimizes the following loss function, which combines reconstruction error and Kullback-

Leibler divergence (KL divergence):

$$\mathcal{L}_{VAE} = -\mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}[\log p(\boldsymbol{x}|\boldsymbol{z})] + D_{KL}(q(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})) = \mathcal{L}_{llike} + \mathcal{L}_{prior} \quad (4)$$

Where the first term represents the reconstruction error and the second term ensures the latent space $\boldsymbol{z}$ follows a prior distribution $p(\boldsymbol{z})$, typically Gaussian. D_{KL} denotes the Kullback-Leibler divergence.

2.1.4. Generative Adversarial Network (GAN) Module

GAN consists of a generator G and a discriminator D. The generator learns the distribution of the data by confronting the two models with each other. In the course of adversarial iterative training, a noisy $\boldsymbol{z}$ is generated from the distribution $p_z(z)$ and supplied to the generator, yielding the output x', where the generator's objective is to align the distribution of the generated data $p_g(x)$, with that of the real data, $p_{data}(x)$. Simultaneously, the discriminator processes input data from both the generator and real sources, yielding an output representing the probability of real data. The discriminator aims to adeptly distinguish between data generated by the true generator and actual data. Within the optimization framework of a Two-player Game, both networks continually refine their predictive capabilities, striving for equilibrium. The goal of GAN is to maximise/minimise the competitive training process:

$$\mathcal{L}_{GAN} = \min_{G}\max_{D} \mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z}|\boldsymbol{x})}[\log D(\boldsymbol{z})] + \mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))] \quad (5)$$

2.2. Classical Anomaly Detection Algorithms

In this subsection, we briefly explain the procedures of unsupervised anomaly detection.

2.2.1. LSTM based anomaly detection algorithm

The algorithm for detecting anomalies based on LSTM is trained on normal time series that is free of anomaly (or with only a negligible portion of anomalous samples). It focuses on detecting anomalies by comparing residuals between predicted and actual values against dynamically set upper and lower thresholds, denoted as ε_{up} and ε_{low}, respectively. Instances exceeding these thresholds are identified as anomalous. Our approach employs a layer of LSTM for anomaly detection. The process for detecting anomalies in the nth moment X_n is as follows: the network is fed with the information of its history of $n-1$ moments, and after passing the last LSTM unit state through the fully connected layer, the predicted information $X_n{}'$ for the nth moment is obtained. The subsequent step involves evaluating the residual between the predicted and actual values at the current moment. After accumulating residuals over a defined period, we use historical residual values along with the N-sigma law to establish the current residual range. This methodical approach enables us to effectively detect anomalies within the time series data. The threshold calculation formula is outlined as follows.

$$\varepsilon_{up} = u + n_{up}\sigma \ , \varepsilon_{low} = u + n_{low}\sigma \quad (6)$$

where u denotes the mean of the historical residuals and σ denotes the variance of the historical residuals. Figure 3 shows the LSTM-based anomaly detection.

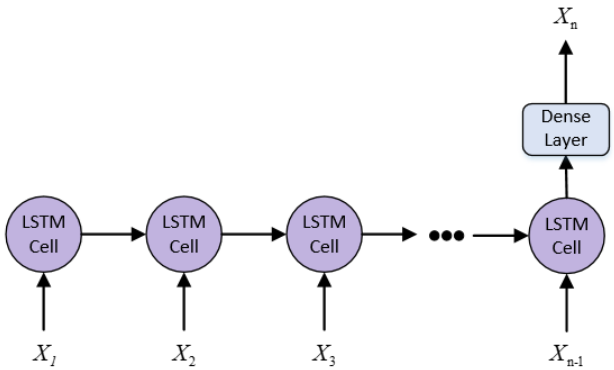

Figure 3 The overall architecture of the LSTM-based anomaly detection network.

2.2.2. LSTM-AE based anomaly detection algorithm

The LSTM-based anomaly detection model for AE learns state transitions between normal sequences and potential space via encoder and decoder. Since the network doesn't understand the potential space representation corresponding to anomalous sequences, the reconstruction error becomes significantly large when processing an anomalous sequence. In our approach, both the encoder and decoder are structured with a single layer of LSTM. The specific flow of performing anomaly detection is as follows: for the nth moment of information X_n to be detected, with the history of $n-1$ moments forming the input $\mathbf{X} = \{X_1, X_2, \ldots, X_n\}$, after passing through the network we obtain the reconstructed sequence $\mathbf{X}' = \{X_1', X_2', \ldots, X_n'\}$, and thus the reconstruction error $\boldsymbol{\gamma} = \{\gamma_1, \gamma_2, \ldots, \gamma_n\}$ between the original sequence and the reconstructed sequence. The history of $n-1$ residuals $\{\gamma_1, \gamma_2, \ldots, \gamma_{n-1}\}$ is also counted using the N-sigma law. The dynamic threshold for the current moment is thus determined and the threshold calculation is shown in Equation 6. If the current residual γ_n exceeds the threshold, it is identified as an anomaly. Figure 4 depicts the structure of the anomaly detection network based on LSTM-AE.

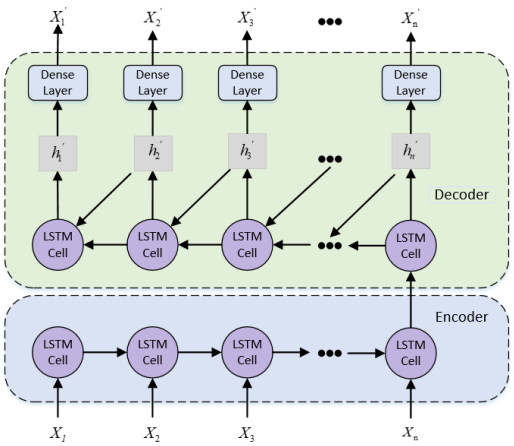

Figure 4 Structure of the LSTM-AE anomaly detection model.

3. PROPOSED METHOD

In this section, we explain our proposed anomaly detection algorithm based on the GAN-VAE model.

3.1. GAN-VAE Model

The GAN-VAE model is constructed by combining GAN and VAE networks. The whole network consists of three components: encoder, decoder and discriminator, where the decoder also acts as the generator of the GAN network. The adversarial training between the generator and the discriminator of the GAN network allows the generator to capture the probability distribution of normal data from a potentially noisy distribution. Given this, the primary framework of the anomaly detection network remains the VAE. The discriminator in the GAN contributes by assisting the decoder in the VAE network (acting as the generator in the GAN network) to enhance the reconstruction of raw data. The encoder, decoder and discriminator of the proposed GAN-VAE network use the GRU structure, taking advantage of the time series modelling of the GRU unit.

Figure 5 illustrates the network structure of GAN-VAE. In our GAN-VAE anomaly detection process, only the encoder and decoder of the model are needed, and the role of the discriminator is mainly to help the decoder to better reconstruct the time series. It should be noted that the state of the first GRU unit of the decoder is randomly initialized within the network, and the state of the subsequent GRU units is passed on by the GRU unit of the previous moment, so the decoder reconstructs the first few time steps, the state of the GRU unit might not have been fully activated. At this time, the decoded information of these time steps will be rounded off. Consequently, the encoder accepts n time steps of information and the decoder decodes m time steps of information ($m > n$), and the decoder focuses only on the latter n time steps of information during the subsequent training and anomaly detection process.

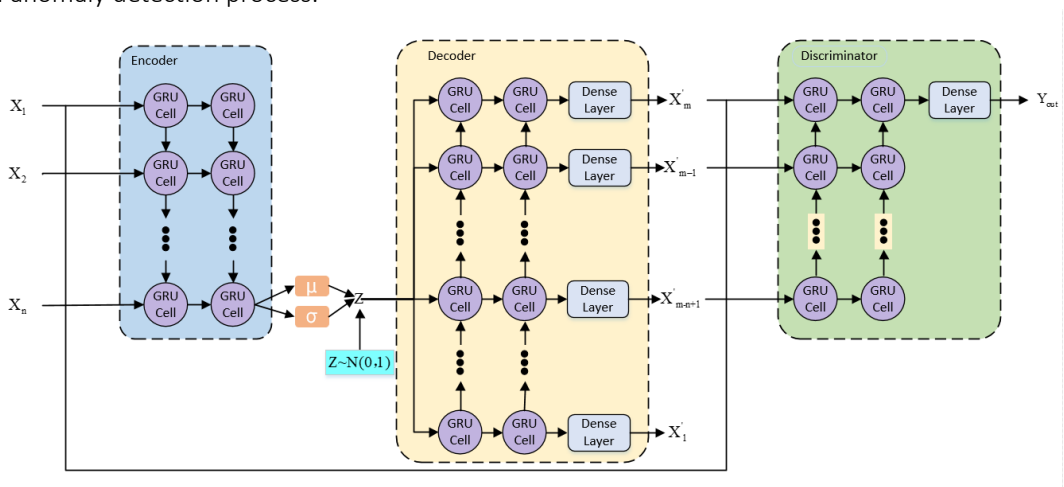

Figure 5 Structure of the GAN-VAE.

3.2. Training process

To accelerate the training of the GAN-VAE model, each module adopts a distributed training method. In addition, we incorporate Wasserstein GAN (WGAN) (Arjovsky et al., 2017) to improve the learning stability of the GAN network and reduce pattern crash problems. We then provide a detailed description of the inputs, outputs, and loss functions for each module in the distributed training process of the GAN-VAE network. For the encoder part, the input comprises only the real-time series $\boldsymbol{X}$ from $p_{data}(x)$, and the output is the compressed latent vector $\boldsymbol{z}$ (where $\boldsymbol{z}$ conforms to a normal distribution). The loss function includes both the D_{KL} and the reconstruction error.

$$\mathcal{L}_{enc} = \mathcal{L}_{llike} + \mathcal{L}_{prior} \tag{7}$$

For the decoder part, the input is the potential vector $\boldsymbol{z}$ compressed by the encoder and the random noise $\boldsymbol{z_p}$ from $p_z(z)$. The output includes the reconstructed sequence $\boldsymbol{X'}$ of the real sequence $\boldsymbol{X}$ and the data $\boldsymbol{X_p}$ generated by the random noise $\boldsymbol{z_p}$. Its loss function includes both the reconstruction error and the cross-entropy loss of the discriminator.

$$\mathcal{L}_{dec} = \mathcal{L}_{llike} + \mathcal{L}_{GAN} \quad (8)$$

For the discriminator part, the inputs are the real data $\boldsymbol{X}$ from $p_{data}(x)$, the reconstructed data $\boldsymbol{X'}$ according to VAE, and the data $\boldsymbol{X_p}$ generated by the random noise $\boldsymbol{z_p}$. The outputs aim to discriminate as accurately as possible between the real data $\boldsymbol{X}$ and the generated sequences $\boldsymbol{X'}$and $\boldsymbol{X_p}$. Its loss function only includes the cross-entropy loss of the discriminator. Throughout the training process, in each batch within each epoch, we train first the encoder, then the decoder, and lastly the discriminator. Alg. 1 gives an overview of the training process.

Algorithm 1: Training GAN-VAE model

initialize network parameters: $\boldsymbol{\theta_{Enc}}, \boldsymbol{\theta_{Dec}}, \boldsymbol{\theta_{Dis}}$

repeat until deadline:

$\mathbf{X} \leftarrow$ random mini-batch from training dataset

// update Encoder model

$\mathbf{Z} \leftarrow \mathrm{Enc}(\mathbf{X})$

$\mathcal{L}_{\boldsymbol{prior}} \leftarrow D_{KL}(q(\mathbf{Z}|\mathbf{X})||p(\mathbf{Z}))$

$\mathbf{X}' \leftarrow \mathrm{Dec}(\mathbf{Z})$

$\mathcal{L}_{\boldsymbol{llike}} \leftarrow -\mathbb{E}_{q(\mathbf{Z}|\mathbf{X})}[\log p(\mathbf{X}|\mathbf{Z})]$

$\boldsymbol{\theta_{Enc}} \overset{+}{\leftarrow} -\nabla_{\boldsymbol{\theta_{Enc}}}(\mathcal{L}_{\boldsymbol{prior}} + \mathcal{L}_{\boldsymbol{llike}})$

// update Decoder model

$\mathbf{Z} \leftarrow \mathrm{Enc}(\mathbf{X})$

$\mathbf{X}' \leftarrow \mathrm{Dec}(\mathbf{Z})$

$\mathcal{L}_{\boldsymbol{llike}} \leftarrow -\mathbb{E}_{q(\mathbf{Z}|\mathbf{X})}[\log p(\mathbf{X}|\mathbf{Z})]$

$\mathbf{Z}_{\boldsymbol{p}} \leftarrow$ samples from prior $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$

$\mathbf{X}_{\boldsymbol{p}} \leftarrow \mathrm{Dec}(\mathbf{Z}_{\boldsymbol{p}})$

$\boldsymbol{\theta_{Dec}} \overset{+}{\leftarrow} -\nabla_{\boldsymbol{\theta_{Dec}}}(\mathcal{L}_{\boldsymbol{llike}} + \log Dis(\mathbf{X}') + \log Dis(\mathbf{X}_{\boldsymbol{p}}))$

// update Discriminator model

$\mathbf{Z} \leftarrow \mathrm{Enc}(\mathbf{X})$

$\mathbf{X}' \leftarrow \mathrm{Dec}(\mathbf{Z})$

$\mathbf{Z}_{\boldsymbol{p}} \leftarrow$ samples from prior $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$

$\mathbf{X}_{\boldsymbol{p}} \leftarrow \mathrm{Dec}(\mathbf{Z}_{\boldsymbol{p}})$

$\boldsymbol{\theta_{Dis}} \overset{+}{\leftarrow} -\nabla_{\boldsymbol{\theta_{Dis}}}(\log Dis(\mathbf{X}) + \log\big(1 - Dis(\mathbf{X}')\big) + \log\Big(1 - Dis\big(\mathbf{X}_{\boldsymbol{p}}\big)\Big))$

3.3. GAN-VAE based anomaly detection

The GAN-VAE based anomaly detection algorithm involves three processes: data processing, sliding window partitioning, and execution detection. The purpose of data processing is to fill in any missing values in the data. Sliding window division processes the time series into fixed-length subsequences, which are then used for subsequent anomaly detection and simulate a real-time detection process. Finally, the execution detection process performs anomaly detection on these divided subsequences.

In the anomaly detection process utilizing our GAN-VAE model, the core components involved are the encoder and the decoder. The detection methodology unfolds as follows: given the sequence data of nth timestamp to be analyzed, denoted as X_n, with the historical data from $n-1$ timestamps forming the input $\mathbf{X} = \{X_1, X_2, \dots, X_n\}$. The encoder processes this input to derive a latent representation $\boldsymbol{z}$ of the real data's potential space. The decoder then reconstructs the sequence, yielding $\mathbf{X}' = \{X_1', X_2', \dots, X_n'\}$. The reconstruction error, denoted by $\boldsymbol{\gamma} = \{\gamma_1, \gamma_2, \dots, \gamma_n\}$, is calculated by comparing the original sequence $\mathbf{X}$ with the reconstructed sequence $\mathbf{X}'$.

Aligning with the adaptive threshold mechanism utilized in LSTM-AE, our approach involves applying the N-sigma rule to ascertain the historical residuals $\{\gamma_1, \gamma_2, \dots, \gamma_{n-1}\}$. This facilitates the determination of dynamic thresholds, ε_{up} and ε_{low}, for the current timestamp, as elaborated in Equation 6. An anomaly is identified at the point of detection when the residual γ_n exceeds these established thresholds, ε_{up} and ε_{low}.

4. Numerical Results

4.1. Parameter setting

In this section, we evaluate the performance of the proposed GAN-VAE model for anomaly detection. The evaluation includes a comparison with the baseline algorithms described in Section II, including LSTM, LSTM-AE, and ARIMA. Table 1 details the hyperparameters of the models, along with the parameters used in model training. The batch size was fixed at 32. The models were constructed using pytorch-1.10 and python-3.7 platforms. Normalization of data was conducted prior to training the models, and a similar normalization process was applied when performing anomaly detection.

Table 1 The hyper parameters of the models considered as well as the parameters for training the models.

Parameters	LSTM	LSTM_AE		GAN-VAE		
		Encoder	Decoder	Encoder	Decoder	Discriminator
Depth	1	1	1	2	2	2
Input Length	96	96		96		
Dimension of Z	none	20		20		
Hidden neural	12	20	20	20,40	40,20	12,6
Learning rate	0.0003	0.0003	0.0003	0.0003	0.0003	0.003
Optimizer	Adam	Adam	Adam	RMSProp	RMSProp	RMSProp

When performing anomaly detection, a longer period needs to be reconstructed in order to capture anomalous change points. However, the conventional input method of GRU, which takes sequential data as a long input vector, may not able to capture the periodic changes in the time series. Inspired by the stacking input method introduced in (Gong et al., 2022), where data points of the same day are aggregated into a vector and fed into a time step of the GRU network, we modify the GRU time step to represent a day rather than the original time granularity of the sequence. This stacking input is applied to both the GAN-VAE model and the baseline models.

An important aspect highlighted in Section 3.1 is that the initial state of the decoder in our GAN-VAE model is internally initialized randomly by the network. The encoder processes n steps of information, while the decoder decodes m time steps of information. According to Table 1, the model's input is 96, achieved by stacking the data points of the same day into one vector. So n is set to 4 steps and we set m to 8 steps. In our experiments, the dynamic thresholds are all determined using the N-sigma law, and the threshold parameters for each method are shown in Table 2.

Table 2 The threshold parameter for the model to perform anomaly detection

Parameters	LSTM	ARIMA	LSTM-AE	GAN-VAE
n_{up}	3.1	4.2	3.5	3.5
n_{low}	3.1	4	3	3

4.2. Data Set Description

In our experiments, we evaluate the anomaly detection performance using two datasets: Base Station (BS) load data and Yahoo data (Laptev & Amizadeh, 2015). Anomalous samples in the dataset are labelled as positive, while normal samples are labelled as negative.

The BS load data is collected by a cellular service provider in response to the total number of BS Radio Resource Control (RRC) connection attempts for BS load Counters. This dataset comprises time series data of individual counters, spanning from 15 January 2018 to 8 July 2018, gathered from over 2,700 locations. With only 0.071% missing points in the raw data, we utilized linear interpolation to fill in the gaps. Preliminary data analyses reveal that the base station data exhibits a 24-hour or 1-week cycle, likely influenced by human activity, as shown in Figure 6. It is worth noting that the characteristics of weekdays and weekends are different in this type of data. Therefore, we need to process them separately and use them together as training data for the model.

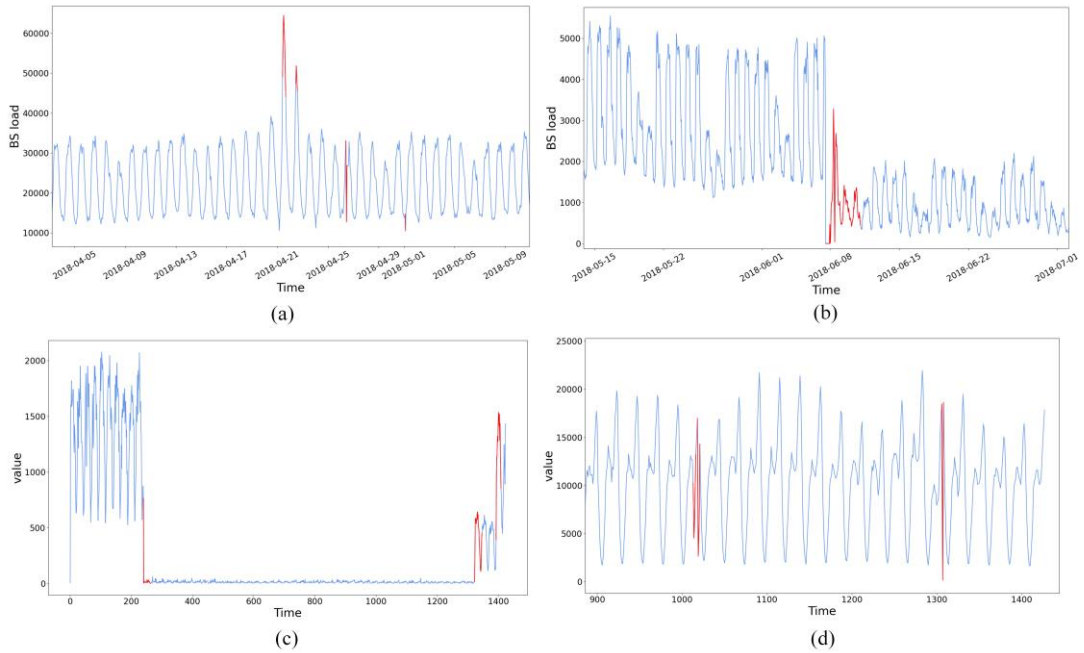

Figure 6 Time series used in our experiment, and the red parts are anomalies. (a)(b) Time series in BS load dataset. (c)(d) Time series in Yahoo dataset.

The Yahoo dataset, released by Yahoo Labs, encompasses both real and synthetic time series. The synthetic dataset contains time series with random seasonality, trend and noise. Within this set, A1Benchmark represents the real data, while the other three segments constitute synthetic data. A1 Benchmark comprises 94,866 points and 1,669 anomalies, resulting in an anomaly rate of 1.76%. We use A1 Benchmark to evaluate the performance of the anomaly detection model, as shown in Figure 6.

The datasets used in the experiments include base station load data and the Yahoo dataset, which cover a wide range of data types. Both datasets contain periodic and non-periodic data, as well as three different types of anomalies: outlier anomalies, contextual anomalies, and collective anomalies. Therefore, the data used in the experiments are representative of large-scale time series data, ensuring the broad generalizability of our proposed algorithm.

In the data processing phase, we first use linear interpolation to fill in the missing values in the data. Next, a sliding window is applied to divide the data into samples of a smaller length. Finally, each item of data is normalized to the range [0,1] using the min-max normalization method. Specifically, for the BS load data, we randomly selected 300 locations as the training set, and the rest of the data were similarly randomly screened from 300 locations as the test set. Regarding the Yahoo data, given its smaller size, we split the time series into two parts, designating one as the training set and the other for testing. The training data from both datasets are collectively utilized to train the GAN-VAE network. It's worth noting that anomalies are excluded from the training data, as our model aims to learn the distribution of normal data.

4.3. Experimental Results

Given the low proportion of anomalies in the data, relying solely on precision is insufficient to evaluate anomaly detection performance. Therefore, recall and F-score are adopted as additional metrics to assess system performance (Chicco & Jurman, 2020). These metrics are defined as follows:

$$Precision = \frac{TP}{TP + FP}$$
$$Recall = \frac{TP}{TP + FN} \quad (9)$$
$$F - score = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

where TP represents the number of correctly detected anomalies, FP represents the number of normal points incorrectly identified as anomalies, and FN represents the number of anomalies incorrectly identified as normal points.

Table 3 shows the performance of our GRU-based GAN-VAE network compared to the baseline anomaly detection method described in Section II. The results demonstrate that the proposed GAN-VAE method outperforms other approaches on both the base station load and Yahoo datasets.

Table 3 Precision, recall and F1 scores of three different classes of time series anomaly detection methods and our GRU-based GAN-VAE method.

Dataset	Method	Precision	Recall	F-score
BS load	LSTM	0.778	0.838	0.807
	ARIMA	0.786	0.647	0.710
	LSTM-AE	0.495	0.177	0.261
	GAN-VAE	**0.861**	**0.902**	**0.881**
Yahoo	LSTM	0.889	0.848	0.868
	ARIMA	0.897	0.873	0.885
	LSTM-AE	0.662	0.538	0.594
	GAN-VAE	**0.900**	**0.877**	**0.888**

4.4. Visual Analysis

The proposed GAN-VAE network is designed to enable the generator to capture the distribution of normal data through mutual confrontation between the GAN's discriminator and the GAN's generator (the decoder of the VAE). This collaborative training enhances the generator's proficiency in reconstructing data with precision. Furthermore, the network learns the distribution of the data on the normal data set. This ensures that after training, when performing anomaly detection, the reconstruction error is small for normal data and large for anomalous data. In order to observe this intuitively, we show the input sequence and the reconstructed sequence when we perform anomaly detection in Figure 7.

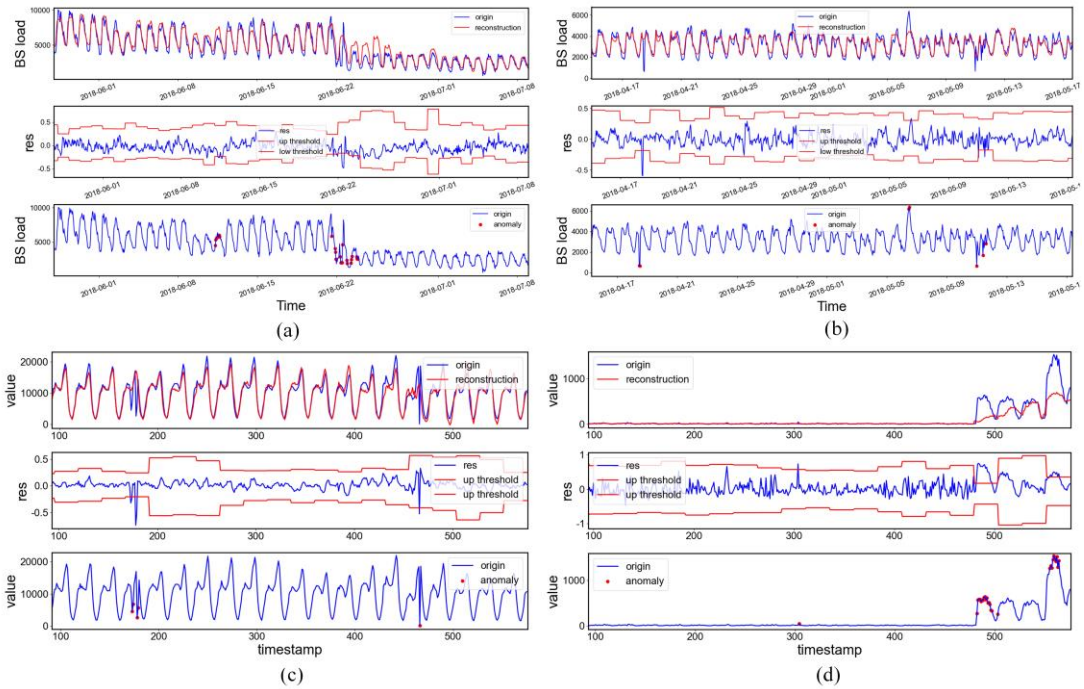

Figure 7 Visualization plots of anomaly detection in the experiment, with the bottom subplot of each plot showing the anomaly detection results. (a)(b) Time series in the BS load dataset. (c)(d) Time series in the Yahoo dataset.

Figure 7 illustrates a schematic representation of the detection process employed by our proposed methodology. In each figure, the upper plot visualizes the original time series as depicted by a blue line, contrasted with the reconstructed sequence shown by a red line. The intermediate plot highlights the discrepancies between the original and reconstructed series, represented by a blue line, while the dynamically computed anomaly detection thresholds, ε_{up} and ε_{low}, are indicated by a red line. Anomalies are identified in instances where the discrepancies surpass these thresholds. The lower plot delineates the results of the anomaly detection process. The performance of our model is notably better in identifying anomalies within large-scale time series data, as evidenced by the schematic.

As shown in Figure 7(a), the data around June 22 demonstrates that the GAN-VAE model can fit the data at the point of interest by combining it with the historical data state. It then identifies a downward collective anomaly based on the reconstruction error between the reconstructed data and the original sequence. Meanwhile, prior to June 11, the sequence data was in a higher-value state on weekdays. However, on June 11, the data suddenly decreased on weekdays, and the GAN-VAE algorithm detected this anomalous state based on the weekday state of the historical data. The detection principles for Figures 7(b) and 7(c) are similar. In Figure 7(d), the initial data values are very low, and then two jumps occur. The algorithm detects these two jumps sequentially based on the historical state of the sequence.

5. CONCLUSIONS

Anomaly detection can promptly identify abnormal behavior in a system to prevent potential failures or security breaches. For instance, in the financial sector, anomaly detection can identify unusual transactions and prevent fraudulent activities. In the manufacturing industry, it can monitor the operating status of equipment and detect failures early, thereby reducing production line downtime

and maintenance costs. In medicine and healthcare, anomaly detection can monitor patients' physiological parameters, detecting abnormalities early to ensure patient health and safety.

This study introduced a sophisticated model for detecting anomalies in large-scale time series, utilizing a GRU-based GAN-VAE framework. The model detects anomalies by examining reconstruction errors of time-series against dynamically adjusted thresholds. It incorporates a GAN to better represent the distributions of normal data and employs distributed and improved WGAN training techniques to enhance efficiency and prevent pattern collapse. The approach of stacking multiple time sequences reduces network time steps, fostering superior sequence pattern learning. Comparative evaluations with benchmark models on diverse datasets validate the superior anomaly detection capabilities of the proposed method.

Our proposed anomaly detection algorithm for GAN-VAE can achieve high accuracy without relying on data labeling. However, the algorithm has certain limitations that require further research and improvement. Specifically, the GRU unit used in the anomaly detection algorithm is only applicable to data sampled at fixed time intervals, which makes the algorithm unsuitable for data sampled at random time intervals.

Acknowledgements

This work was supported by the Science and Technology Project of State Grid Corporation of China (SGJSWX00KJJS220847).

References

Arjovsky, M., Chintala, S., & Bottou, L. (2017). Wasserstein Generative Adversarial Networks. In *International Conference on Machine Learning*, 214–223.

Chen, X., Lin, X., Li, Z., & Fan, H. (2023). Unsupervised Time Series Anomaly Detection Based on Adversarial Interpolation and Pseudo-anomaly Calibration. *2023 9th International Conference on Control, Automation and Robotics (ICCAR)*, 91-95. https://doi.org/10.1109/ICCAR57134.2023.10151775

Chicco, D., & Jurman, G. (2020). The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genomics, 21*(1). https://doi.org/10.1186/s12864-019-6413-7

Cui, W., & Wang, H. (2017). Anomaly detection and visualization of school electricity consumption data. *2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA)*, 606-611. https://doi.org/10.1109/ICBDA.2017.8078707

Geiger, A., Liu, D., Alnegheimish, S., Cuesta-Infante, A., & Veeramachaneni, K. (2020). TadGAN: Time Series Anomaly Detection Using Generative Adversarial Networks. *2020 IEEE International Conference on Big Data (Big Data)*, 33-43. https://doi.org/10.1109/BigData50022.2020.9378139

Gong, X., Liao, S., Hu, F., Hu, X., Liu, C. J. I. A. P. C. o. C., & Systems. (2022). Autoencoder-Based Anomaly Detection for Time Series Data in Complex Systems. *2022 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, 428-433. https://doi.org/10.1109/APCCAS55924.2022.10090260

Laptev, N., & Amizadeh, S. (2015). *Yahoo anomaly detection dataset s5*. http://webscope.sandbox.yahoo.com/catalog.php.

Liu, D., Pang, J., Xu, B., Liu, Z., Zhou, J., & Zhang, G. (2017). Satellite Telemetry Data Anomaly Detection with Hybrid Similarity Measures. *2017 International Conference on Sensing, Diagnostics, Prognostics, and Control (SDPC)*, 591-596. https://doi.org/10.1109/SDPC.2017.116

Liu, J., Bai, X., Wang, M., Tuarob, S., & Xia, F. (2024). Anomalous citations detection in academic networks. *Artificial Intelligence Review, 57*(4). https://doi.org/10.1007/s10462-023-10655-5

Niu, Z., Yu, K., & Wu, X. (2020). LSTM-Based VAE-GAN for Time-Series Anomaly Detection. *Sensors* (Basel, Switzerland), 20. https://doi.org/10.3390/s20133738

Qin, G., Chen, Y., & Lin, Y.-X. (2018). Anomaly Detection Using LSTM in IP Networks. *2018 Sixth International Conference on Advanced Cloud and Big Data (CBD)*, 334-337. https://doi.org/10.1109/CBD.2018.00066

Tron, T., Resheff, Y. S., Bazhmin, M., Weinshall, D., & Peled, A. (2018). ARIMA-based motor anomaly detection in schizophrenia inpatients. *2018 IEEE EMBS International Conference on Biomedical & Health Informatics (BHI)*, 430-433. https://doi.org/10.1109/BHI.2018.8333460

Wang, Z., Pei, C., Ma, M., Wang, X., Li, Z., Pei, D., ... & Xie, G. (2024, May). Revisiting VAE for Unsupervised Time Series Anomaly Detection: A Frequency Perspective. In *Proceedings of the ACM on Web Conference 2024* (pp. 3096-3105).

Xu, S., Liu, H., Duan, L., & Wu, W. (2021). An Improved LOF Outlier Detection Algorithm. *2021 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA)*. https://doi.org/10.1109/icaica52286.2021.9498181

Xu, H., Wang, Y., Jian, S., Liao, Q., Wang, Y., & Pang, G. (2024). Calibrated one-class classification for unsupervised time series anomaly detection. *IEEE Transactions on Knowledge and Data Engineering*.

Zavrak, S., & Iskefiyeli, M. (2020). Anomaly-Based Intrusion Detection From Network Flow Features Using Variational Autoencoder. *IEEE Access, 8*, 108346-108358. https://doi.org/10.1109/ACCESS.2020.3001350

Zhu, G., Zhao, H., Liu, H., & Sun, H. (2019). A Novel LSTM-GAN Algorithm for Time Series Anomaly Detection. *2019 Prognostics and System Health Management Conference (PHM-Qingdao)*, 1-6. https://doi.org/10.1109/phm-qingdao46334.2019.8942842

Zehra, S., Faseeha, U., Syed, H. J., Samad, F., Ibrahim, A. O., Abulfaraj, A. W., & Nagmeldin, W. (2023). Machine Learning-Based Anomaly Detection in NFV: A Comprehensive survey. *Sensors, 23*(11), 5340. https://doi.org/10.3390/s23115340

Zhu, T., Li, P., Yu, L., Chen, K., & Chen, Y. (2020). Change Point Detection in Dynamic Networks Based on Community Identification. *IEEE Transactions on Network Science and Engineering, 7*, 2067-2077. https://doi.org/10.1109/TNSE.2020.2973328