Information Sciences 676 (2024) 120852

Contents lists available at ScienceDirect

Information Sciences

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

VAEAT: Variational AutoeEncoder with adversarial training for multivariate time series anomaly detection

Sheng He [a], Mingjing Du [a,*], Xiang Jiang [a], Wenbin Zhang [a,b], Congyu Wang [a]

[a] School of Computer Science and Technology, Jiangsu Normal University, Xuzhou, 221116, China
[b] College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao, 266590, China

ARTICLE INFO

Keywords:
Multivariate time series
Anomaly detection
Variational autoencoder
Adversarial training

ABSTRACT

High labor costs and the requirement for significant domain expertise often result in a lack of anomaly labels in most time series. Consequently, employing unsupervised methods becomes critical for practical industrial applications. However, prevailing reconstruction-based anomaly detection algorithms encounter challenges in capturing intricate underlying correlations and temporal dependencies in time series. This study introduces an unsupervised anomaly detection model called Variational AutoeEncoder with Adversarial Training for Multivariate Time Series Anomaly Detection (VAEAT). Its fundamental concept involves adopting a two-phase training strategy to improve anomaly detection precision through adversarial reconstruction of raw data. In the first phase, the model reconstructs raw data to extract its basic features by training two enhanced variational autoencoders (VAEs) that incorporate both the long short-term memory (LSTM) network and the attention mechanism in their common encoder. In the second phase, the model refines reconstructed data to optimize the reconstruction quality. In this manner, this two-phase VAE model effectively captures intricate underlying correlation and temporal dependencies. A large number of experiments are conducted to evaluate the performance on five publicly available datasets, and experimental results illustrate that VAEAT exhibits robust performance and effective anomaly detection capabilities. The source code of the proposed VAEAT can be available at https://github.com/Du-Team/VAEAT.

1. Introduction

As the accumulation of a vast amount of time series within enterprises and organizations, time series [1] anomaly detection has become a critical technique for identifying exceptional situations [2], predicting future events [3], and maintaining operations [4]. In time series data, anomalous events exhibit a clear distinction from normal events along the time axis, as depicted in Fig. 1. The objective of anomaly detection [5] is to identify data points that deviate from the expected behavior. These data points potentially indicate faults, anomalous events, or other irregular situations [6]. Due to the fact that most time series data present a large number and high dimension [7], the data are often difficult to be labeled for this reason [8]. Therefore, studies on multivariate time series [9,10] anomaly detection predominantly center on unsupervised approaches [11].

* Corresponding author.
E-mail addresses: hes@jsnu.edu.cn (S. He), dumj@jsnu.edu.cn (M. Du), xjiang@jsnu.edu.cn (X. Jiang), zwbwen@jsnu.edu.cn (W. Zhang), wangcongyu@jsnu.edu.cn (C. Wang).

https://doi.org/10.1016/j.ins.2024.120852
Received 18 February 2024; Received in revised form 19 April 2024; Accepted 29 May 2024

Available online 3 June 2024

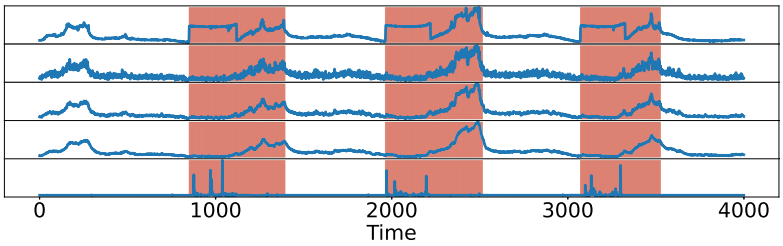


Fig. 1. Multivariate time series with anomaly points. The shaded area refers to the interval where the anomaly is labeled.

In recent years, advancements in related technologies have resulted in the emergence of numerous unsupervised methods based on deep learning [12], including WANEH [13] and DeepLog [14]. Among them, variational autoencoder (VAE)-based methods are extensively employed for their strong theory foundation and reliable stability. However, they have limitations, notably in grappling with the intricacies learning the underlying data distribution. Researchers propose some solutions to overcome this challenge. An intuitive strategy is to use more complex prior distributions, which can effectively capture the intricate patterns and structures in the data. For example, Liao et al. [15] use the Gaussian mixture model as a prior for the distribution of multivariate time series. This helps capture the complex characteristic and multimodal distribution of time series. Rezende et al. [16] map the simple probability distribution (such as the standard normal distribution) to a complex one through a series of asymptotic transformations. This method can be used to improve approximate inference in variational inference, making the model better approximate the true posterior distribution. However, these methods require prior knowledge or assumptions about the underlying variables. If these assumptions differ from the actual data distribution, they could constrain the expressiveness of the model or induce a decline in performance. In addition, they frequently ignore dependencies within time series, especially long-term dependencies, which is also an issue that needs to be addressed.

For solving the aforementioned problems, a novel unsupervised anomaly detection method for multivariate time series, VAEAT, is proposed. The method mainly adopts the architecture of VAE, including a single encoder and two decoders. A two-stage training with the idea of adversarial training is adopted, enhancing the capability of extracting complex underlying distributions of raw data and effectively mitigating the limitations of VAE. In addition, we use LSTM in the encoder, which enables better access to the dependencies between time series. Finally, we solve the problem of forgetting past information during the feature extraction of long sequences by introducing an attention mechanism in LSTM.

The main contributions of this paper can be generalized in the following points:

- We propose VAEAT, an approach for unsupervised anomaly detection on multivariate time series. This method demonstrates the strong feature extraction and noise resistance for time series data, resulting in improving its capability of distinguishing abnormal and normal data.
- We develop an improved variational autoencoder architecture that integrates an LSTM module and an attention mechanism in its encoder, and performs data reconstruction using one pair of structurally identical decoders. The proposed structure intends to better capture dependencies within time series.
- We use a two-phase training strategy. By introducing different regularization terms for the two decoders, the idea of adversarial training is incorporated in the second phase. This strategy helps to discover complex data structures while enhancing the noise resistance to the model.
- Experimental outcomes across five different datasets indicate a significant improvement over the previous methods, positioning VAEAT as a novel baseline.

The remaining sections in the paper are organized in the following way. Related methods used for anomaly detection on multivariate time series are explored in Section 2. The comprehensive summary on the method we propose is given in Section 4. The detail presentation on our experiment outcomes is given in Section 5. Finally, Section 6 provides a conclusion encompassing the entirety of this paper.

2. Related work

Anomaly detection in time series can be classified into two categories: unsupervised and supervised algorithms [17]. This paper focus on the unsupervised scenarios. Unsupervised anomaly detection methods, according to their objectives, are further classified as two groups: the forecasting-based and the reconstruction-based methods.

The Forecasting-Based Methods. An fundamental concept of the methods is to employ a prediction algorithm to forecast future values for target variables. This is done by comparing predicted values and actual recorded values when new records are available. If the residual error surpasses a specific threshold, it is considered as an anomaly. To address the problem that traditional anomaly detection methods may not be able to effectively capture complex seasonal features, AD-LTI [18] takes into account multiple seasonal patterns in time series data to effectively capture anomalies on different time scales. However, it ignores the fact that datasets are often highly imbalanced. To address this problem, Chen et al. [19] propose an anomaly detection method based on classical echo state network (ESN), called HealthESN. However, the two methods emphasize the dependencies of the time series on time dimensions, but

Table 1
Notations and their descriptions used subsequently.

Notations	Descriptions
$\boldsymbol{W}_f, \boldsymbol{W}_C, \boldsymbol{W}_i, \boldsymbol{W}_o$	The weight matrices
$\boldsymbol{b}_f, \boldsymbol{b}_C, \boldsymbol{b}_i, \boldsymbol{b}_o$	The bias terms
$\boldsymbol{X}$	A multivariate time series
$\boldsymbol{x}_t$	The data point at time step t
T	The timestamp length
m	The dimension sizes for data points
$\boldsymbol{y}$	The test result of each data point of the test data
y_t	The test result of a data point at time step t
$\boldsymbol{\epsilon}$	The minor constancy vector
$\boldsymbol{W}_t$	The context window at time step t
K	The context window length
$\boldsymbol{S}_t$	The anomaly scores
$\boldsymbol{Z}$	The latent variable
$\boldsymbol{W}_t^{1'}, \hat{\boldsymbol{W}}_t^{1'}$	The reconstructed data for the first variational autoencoder in the first phase
$\boldsymbol{W}_t^{2'}$	The reconstructed data of the second variational autoencoder in the first phase
$\boldsymbol{W}_t^{2''}, \hat{\boldsymbol{W}}_t^{2''}$	The reconstructed data for the second variational autoencoder in the second phase
e	The training epoch
$\boldsymbol{\mu}_Z, \boldsymbol{\mu}, \boldsymbol{\sigma}_Z^2, \boldsymbol{\sigma}^2$	The mean and variance for distributions of latent variables

neglect those on feature dimensions. To address this problem, MTAD-GAT [20] not only treats an individual characteristic for one of the time series, but also includes a dual-layer graph attention mechanism.

The Reconstruction-Based Methods. An fundamental concept of the methods is to employ a model to learn patterns in normal time series data and subsequently utilize this model to reconstruct newly acquired data. Reconstruction errors are applied to identify anomalies in cases where reconstituted data deviates significantly from raw data. Among them, VAE-based methods are more common. Although VAE has the stable performance, it has the problem of difficulty in terms of its ability to estimate underlying distributions of data. To address this problem, OmniAnomaly [21] employs technologies, including random variable concatenation and plane normalized flow in order to reveal hidden distributions of data through modeling their robust representations. However, it mainly focuses on the relationship of time series on time dimensions while paying less attention towards relationships on feature dimensions. To solve this problem, MSCRED [22] identifies the state of multi-level systems over a wide range of timescales by creating a multi-scale signature matrix. It also uses convolutional encoders for encoding the correlation between features at the same time and a specially-designed network for capturing patterns in time. Unlike this method, MEGA [23] integrates the discrete wavelet transform in an autoencoder. This integration is used for decomposing data into distinct frequency components before their subsequent reconstruction. However, the aforementioned two methods are more sensitive to noise. To solve this problem, Bagel [24] uses conditional variational autoencoder combined with time information and dropout layers. In addition to it, RDSMM [25] includes a new generation model, an inference model and an elaborated specified emission model on the basis of statistical theories. Another one addressing the problem of data noise is generative adversarial network (GAN). TMANomaly [26] is made up of two same sub-networks that train each other by alternately assuming the roles of reconstructor and discriminator. However, it pays less concern for correlations among multivariate time series features. For addressing this problem, AMBi-GAN [27] allows the network structure of generators and discriminators to incorporate the attention mechanism on the LSTM networks. However, the model training is not stable enough due to the own reasons of GAN. With the aim of solving this problem, ALAD [28] employs bidirectional GANs for learning reverse features. This ensures the consistency of data space and latent space loops as well as stabilizes the training of GAN. Furthermore, some researchers combine VAE with GAN. BeatGAN [29] uses the adversarial training idea to regularize the reconstruction error. This addresses the challenge of excessive labeling and data balancing needed for classification, while reducing reliance on regularization during reconstruction. DAEMON [30] builds upon BeatGAN and regularizes the latent space variables using adversarial generation techniques. However, both of the above methods require the involvement of discriminators, making the models more complex.

3. Preparatory work

Before describing the model in detail, we provide an elaborated analysis of several key points mentioned in the model: Variational Autoencoder, Long Short-Term Memory Network, and the attention mechanism. A short definitional description is given for the notations throughout the paper.

3.1. Notation descriptions

In this subsection, we present the notations and their descriptions used subsequently in the paper, as shown in Table 1.

3.2. Variational autoencoder

The operation of VAE is similar to that of an autoencoder (AE), but it represents latent features for time series not as single values but as a range of possible values. D-dimensional input data $\boldsymbol{W}$ passes through the encoder to yield a distribution of k-dimensional latent feature representations. These representations are sampled to obtain the mean $\boldsymbol{\mu}$ and the variance $\boldsymbol{\sigma}^2$ of the latent feature $\boldsymbol{Z}$. Since the sampling process can't be back-propagated during training, the reparameterization trick is applied to combine them, as shown in Eq. (1). After obtaining the latent variables, the decoder is used to generate the reconstructed data $\boldsymbol{W}'$. Finally, we have our objective function:

$$Loss = \|\boldsymbol{W} - VAE(\boldsymbol{W})\|_F^2 + \sum_{i=1}^{n} KL(N(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)||N(0,1)), \tag{1}$$

where $KL(\cdot \parallel \cdot)$ denotes the Kullback-Leibler divergence within two distributions. $N(\cdot,\cdot)$ represents a normal distribution. n is the number of samples. The purpose of $KL(\cdot \parallel \cdot)$ is to make the output of our encoder as close as possible to a Gaussian distribution under the assumption that $\boldsymbol{Z}$ follows a Gaussian distribution.

3.3. Long short-term memory networks

LSTM is a variant of Recurrent Neural Networks (RNNs) designed specifically for processing and modeling time-series data and other data with temporal dependencies. The core components of it include the input gate, the forget gate, the output gate, and the cell state.

Forget Gate. To determine how much of the previous cell state information should be forgotten, we calculate it using the current input $\boldsymbol{w}_t$ and the previous time step's hidden state $\boldsymbol{h}_{t-1}$. We pass these through a fully connected layer and apply the sigmoid function to obtain the value $\boldsymbol{f}_t$ of the forget gate. This value lies between 0 and 1, where 0 means completely forgetting the information, and 1 means retaining it entirely.

$$\boldsymbol{f}_t = sig(\boldsymbol{W}_f \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{w}_t] + \boldsymbol{b}_f), \tag{2}$$

where $\boldsymbol{W}_f$ is the weight matrix, $\boldsymbol{b}_f$ is the bias term, and $sig(\cdot)$ is the sigmoid function.

Input Gate. LSTM needs to determine which new information should be stored in the memory cell. This consists of two parts: one is the input gate, which decides which parts of the memory cell we will update; the other is a tanh layer, which creates a new candidate value vector that may be added to the memory cell. The values $\boldsymbol{i}_t$ for the input gate and the candidate value $\tilde{\boldsymbol{C}}_t$ are both calculated based on the current input and the hidden state from the previous time step. The formula for the input gate is as follows:

$$\tilde{\boldsymbol{C}}_t = \tanh(\boldsymbol{W}_C \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{w}_t] + \boldsymbol{b}_C),\ \boldsymbol{i}_t = sig(\boldsymbol{W}_i \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{w}_t] + \boldsymbol{b}_i), \tag{3}$$

where $\boldsymbol{W}_C$ and $\boldsymbol{W}_i$ are the weight matrixes, $\boldsymbol{b}_C$ and $\boldsymbol{b}_i$ are the bias terms.

Updating the Memory Cell. The memory cell is updated based on the decisions made by the forget gate and the input gate. We multiply the cell state by the value $\boldsymbol{f}_t$ of the forget gate, which means we forget a part of the state information. Then we multiply the value $\boldsymbol{i}_t$ of the input gate by the candidate value $\tilde{\boldsymbol{C}}_t$ and add it to the cell state, representing the addition of some new state information.

$$\boldsymbol{C}_t = \boldsymbol{f}_t \odot \boldsymbol{C}_{t-1} + \boldsymbol{i}_t \odot \tilde{\boldsymbol{C}}_t, \tag{4}$$

where $\odot$ represents the product of the elements in the corresponding positions of the operands on either side of this symbol.

Output gate and hidden state are crucial components in the architecture. The output gate is responsible for determining the appropriate output based on the cell state. By passing both the current input and previous time step's hidden state through a fully connected layer, we can obtain the value of the output gate $\boldsymbol{o}_t$ using the sigmoid function. This value ranges between 0 and 1, where 0 signifies no output while 1 represents full output. Subsequently, applying the tanh function to the cell state yields a result ranging from -1 to 1, which is then multiplied by the output gate value to derive our final hidden state $\boldsymbol{h}_t$.

$$\boldsymbol{o}_t = sig(\boldsymbol{W}_o \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{w}_t] + \boldsymbol{b}_o), \boldsymbol{h}_t = \boldsymbol{o}_t \odot \tanh(\boldsymbol{C}_t), \tag{5}$$

where $\boldsymbol{W}_o$ is the weight matrix, $\boldsymbol{b}_o$ is the bias term, and $\tanh(\cdot)$ is the hyperbolic tangent function.

3.4. Attention mechanism

After the output results of each step of LSTM described above, we impose the attention mechanism so that it can fuse more information of earlier time steps. The specific process is as follows:

$$\boldsymbol{M} = \tanh(\boldsymbol{H}), \tag{6}$$

$$\boldsymbol{\alpha} = softmax(\boldsymbol{\omega}^T \boldsymbol{M}), \tag{7}$$

$$\boldsymbol{r} = \boldsymbol{H}\boldsymbol{\alpha}^T \tag{8}$$

where $\boldsymbol{H}$ denotes the matrix stitched together as the output of each step of LSTM, $\boldsymbol{M}$ is the matrix that undergoes the activation function $\tanh(\cdot)$, $\boldsymbol{\omega}$ is the weight matrix, $\boldsymbol{\alpha}$ is the weight matrix after undergoing $softmax(\cdot)$, and $\boldsymbol{r}$ is the matrix that finally incorporates the past information.

4. Method

In this section, we first delve into the motivation and reasons behind the proposed method of the paper. Subsequently, we provide a description of some basic issues. Then, we describe the preprocessing of the data. Immediately after that, we elaborate on the specifics of the method presented in the paper. Finally, we develop a discussion on anomaly detection.

4.1. Motivation

In this section, we attempt to construct an anomaly detection model based on the VAE framework. VAE is an extension of the autoencoder that incorporates the principles of probabilistic modeling. Within the framework, the encoder embeds input data into a lower dimensional latent space and samples latent variables from it. Subsequently, the decoder reconstructs input data from variables. The basic principles of VAE-based methods are the computation of anomaly scores through the measurement of reconstruction errors from raw data to reconstructed data. Specifically, models are trained using normal data. Consequently, it exhibits excellent reconstructions for normal data (with low anomaly scores) and poor reconstructions for anomalous data (with high anomaly scores). However, if the normal data is very similar to the anomalous data, it can make the anomaly difficult to detect. Two potential solutions to address the problem include magnifying the reconstruction error for anomalous data and incorporating a more robust detection module or mechanism. Increasing the number of training layers may contribute to amplifying the reconstruction error of anomalous data; however, deeper networks may pose challenges in training and stability. Alternatively, using GAN may be a feasible solution to improve model detection. The principle of GAN is to train both generators and discriminators through a competitive process, in which generators aim to generate real data for deceiving discriminators, and discriminators seek towards accurately distinguishing from real data to generated data. However, there are some fatal problems with GAN, such as modal collapse.

Our goal is to increase reconstruction errors on anomalous data and enhance anomaly detection capabilities in the model while maintaining model's simplicities and stabilities. For this purpose, we construct a method named VAEAT, which uses variational autoencoders as the main architecture and creates a two-phase training strategy using adversarial training ideas. Our framework inherits the training stability inherent in the VAE model. Our proposed training strategy imitates the adversarial training idea of the GAN model while circumventing its inherent limitations. These improvements enhance anomaly detection capabilities and increase robustness to noise.

4.2. Problem formulation

Multivariate time series $\boldsymbol{X} = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_T\}$ having timestamp length T is defined [10]. The data point captured in certain time t is $\boldsymbol{x}_t$, $\boldsymbol{x}_t \in R^m$. m denotes dimension sizes for data points [31].

For the trained model, we use an independent testing dataset that has the same origin as the training dataset used during training. Specifically, we employ a testing time series with a length denoted as $\hat{T}$. The overall test result of each data point of the test data is $\boldsymbol{y} = \{y_1, ..., y_{\hat{T}}\}$. The test result of a data point in a specific time t represents $y_t \in \{0, 1\}$, which is used to indicate whether the point is abnormal (1 indicates an anomalous data point).

4.3. Data preprocessing

The data preprocessing phase comprises data standardization. Let us normalize time series $\boldsymbol{X}$ in the following way:

$$\boldsymbol{x}_t \leftarrow \frac{\boldsymbol{x}_t - min(\boldsymbol{X})}{max(\boldsymbol{X}) - min(\boldsymbol{X}) + \boldsymbol{\varepsilon}}, \tag{9}$$

where $\boldsymbol{x}_t$ is the data point, $min(\boldsymbol{X})$ represents a minimal vector of time series, as well as $max(\boldsymbol{X})$ represents a maximal vector. $\boldsymbol{\varepsilon}$ represents the minor constancy vector preventing the single sequence from having a single value that causes the denominator of the above equation to be 0 and thus not be calculable.

Following standardization, we create a context window of length K for each data point $\boldsymbol{x}_t$ in time t. This context window enables the capture of temporal dependencies in time series as inputs to the model, defined as $\boldsymbol{W}_t = \{\boldsymbol{x}_{t-K+1}, ..., \boldsymbol{x}_t\}$. Anomaly score $\boldsymbol{S}_t$ of $\boldsymbol{x}_t$ is computed using $\boldsymbol{W}_t$, and then is used to determine whether $\boldsymbol{x}_t$ is an anomaly [32].

4.4. Unsupervised anomaly detection

VAEAT consists of two variational autoencoders as illustrated by Fig. 2, the first VAE (denoted by V_1) and the second VAE (denoted by V_2). Notably, they share a common encoder, Encoder (denoted as E). The decoders comprise two components: Decoder1 (denoted by D_1) and Decoder2 (denoted by D_2). The outputs of V_1 and V_2 are:

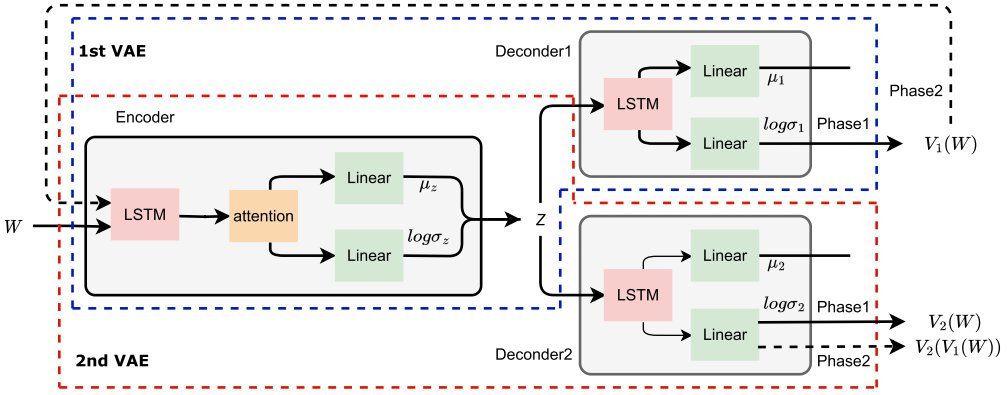


Fig. 2. The VAEAT Model. Phase 1: Raw time series data W can be encoded in latent variables Z through a common encoder (incorporating LSTM with an attention mechanism). Then, reconstructed data can be produced from respective decoders. Phase 2: Reconstructed data produced from V_1 (in the blue box) can be reintroduced as input data into the common encoder, and the encoded latent variable is utilized for generating the final reconstructed data by a decoder in V_2 (in the red box).

$$V_1(\boldsymbol{W}) = D_1(E(\boldsymbol{W})), \tag{10}$$

$$V_2(\boldsymbol{W}) = D_2(E(\boldsymbol{W})) \tag{11}$$

Next, we delve into the details. In the first phase, the window data $\boldsymbol{W}$ is fed into the encoder, passing through the LSTM network and its associated attention mechanism, and after sampling, obtaining the latent variable $\boldsymbol{Z}$. Subsequently, the reconstructed data $V_1(\boldsymbol{W})$ and $V_2(\boldsymbol{W})$, corresponding to our normal window data W, are obtained from $\boldsymbol{Z}$ through Decoder1 and Decoder2 respectively. In the second phase, since we refer to the idea of adversarial training, V_2 is regarded as the discriminator, and the reconstructed data $V_1(\boldsymbol{W})$ in the first phase is discriminated as the generated data. Similarly, V_1 is treated as the generator. That is, the goal of V_1 is to fool V_2 through generations of reconstructed data, while an objective of V_2 is to learn distinguishing if data is real data W or reconstructed data produced from V_1. For specific details, please refer to Algorithm 1. $\boldsymbol{W}_t^{1'}$ represents the reconstructed data for the first variational autoencoder in the first phase, $\boldsymbol{W}_t^{2'}$ denotes reconstructed data of V_2 in the first phase, as well as $\boldsymbol{W}_t^{2''}$ represents the reconstructed data for the second variational autoencoder in the second phase.

Algorithm 1 Training algorithm.

Input: Normal windows Dataset $\mathcal{W} = \{\boldsymbol{W}_1, ..., \boldsymbol{W}_T\}$, number epochs N_{epoch}
Output: Trained V_1, V_2

Initialize weights E, D_1, D_2
$e \leftarrow 1$
while $e \leq N_{epoch}$ **do**
  for $t = 1$ *to* T **do**
    $\boldsymbol{W}_t^{1'}, \boldsymbol{W}_t^{2'} \leftarrow D_1(E(\boldsymbol{W}_t)), D_2(E(\boldsymbol{W}_t))$
    $\boldsymbol{W}_t^{2''} \leftarrow D_2(E(\boldsymbol{W}_t^{1'}))$
    $L_1 \leftarrow \frac{1}{e}(\left\|\boldsymbol{W}_t - \boldsymbol{W}_t^{1'}\right\|_F^2$
    $+ \sum_{i=1}^{n} KL(N(\boldsymbol{\mu}_Z, \boldsymbol{\sigma}_Z^2) \| N(0,1)))$
    $+ (1 - \frac{1}{e}) \left\|\boldsymbol{W}_t - \boldsymbol{W}_t^{2''}\right\|_F^2$
    $L_2 \leftarrow \frac{1}{e}(\left\|\boldsymbol{W}_t - \boldsymbol{W}_t^{2'}\right\|_F^2$
    $+ \sum_{i=1}^{n} KL(N(\boldsymbol{\mu}_Z, \boldsymbol{\sigma}_Z^2) \| N(0,1)))$
    $- (1 - \frac{1}{e}) \left\|\boldsymbol{W}_t - \boldsymbol{W}_t^{2''}\right\|_F^2$
    $E, D_1, D_2 \leftarrow$ update weights using L_1 and L_2
  $e \leftarrow e + 1$

Phase 1: Reconstructing Inpution. In the first phase, we input raw data $\boldsymbol{W}$ into two variational autoencoders to obtain the corresponding reconstructed data. The specific details are as follows: raw data $\boldsymbol{W}$ is inputed into the common encoder for compression, and then the mean and variance distribution of the compression vector is obtained. Subsequently, the latent variable ($\boldsymbol{Z}$) is obtained. It is decoded through two decoders for obtaining corresponding reconstructed data. Therefore, our training objectives in the first phase are as follows:

$$L_1 = \left\|\boldsymbol{W} - V_1(\boldsymbol{W})\right\|_F^2 + \sum_{i=1}^{n} KL(N(\boldsymbol{\mu}_Z, \boldsymbol{\sigma}_Z^2) \| N(0,1)) \tag{12}$$

$$L_2 = \left\| \boldsymbol{W} - V_2(\boldsymbol{W}) \right\|_F^2 + \sum_{i=1}^{n} KL(N(\boldsymbol{\mu}_Z, \boldsymbol{\sigma}_Z^2) \| N(0,1)) \tag{13}$$

where $KL(\cdot \| \cdot)$ denotes the Kullback-Leibler divergence within two distributions. $N(\cdot, \cdot)$ represents a normal distribution.

Phase 2: Adversarial training. We use an approach similar to GAN. We use the reconstructed data $V_1(\boldsymbol{W})$ from the first phase as input to V_2. The goal is to train V_2 as a discriminator for distinguishing between real data and reconstructed data generated by V_1, as well as training V_1 as a generator to generate the reconstructed data so that V_2 mistakenly thinks that reconstructed data is real data. V_1 aims to have a minimum discrepancy from raw data W to the output of V_2 in the second phase, and V_2 aims to have a maximum discrepancy between the two. Therefore, our training objectives in the second phase are as follows:

$$\min_{V_1} \max_{V_2} \left\| \boldsymbol{W} - V_2(V_1(\boldsymbol{W})) \right\|_F^2 \tag{14}$$

The above loss function can be converted to the following losses:

$$L_1 = + \left\| \boldsymbol{W} - V_2(V_1(\boldsymbol{W})) \right\|_F^2 \tag{15}$$

$$L_2 = - \left\| \boldsymbol{W} - V_2(V_1(\boldsymbol{W})) \right\|_F^2 \tag{16}$$

Aggregating the Training Objective. Given that we have separately obtained the reconstruction phase loss and the adversarial training phase loss, we can now use the final loss function to summarize the losses of these two phases:

$$\begin{aligned} L_1 = & \frac{1}{e}(\left\| \boldsymbol{W} - V_1(\boldsymbol{W}) \right\|_F^2 \\ & + \sum_{i=1}^{n} KL(N(\boldsymbol{\mu}_Z, \boldsymbol{\sigma}_Z^2) \| N(0,1))) \\ & + (1 - \frac{1}{e}) \left\| \boldsymbol{W} - V_2(V_1(\boldsymbol{W})) \right\|_F^2 \end{aligned} \tag{17}$$

$$\begin{aligned} L_2 = & \frac{1}{e}(\left\| \boldsymbol{W} - V_2(\boldsymbol{W}) \right\|_F^2 \\ & + \sum_{i=1}^{n} KL(N(\boldsymbol{\mu}_Z, \boldsymbol{\sigma}_Z^2) \| N(0,1))) \\ & - (1 - \frac{1}{e}) \left\| \boldsymbol{W} - V_2(V_1(\boldsymbol{W})) \right\|_F^2 \end{aligned} \tag{18}$$

where e denotes a training epoch, $\boldsymbol{\mu}_Z$ and $\boldsymbol{\sigma}_Z^2$ represent the mean and variance for distributions of latent variables.

4.5. Anomaly detection

Following model training, the trained model is used for detecting the test data containing anomalies, as shown in Algorithm 2. $\hat{\boldsymbol{W}}_t^{1'}$ represents the reconstructed data for the first variational autoencoder in the first phase, and $\hat{\boldsymbol{W}}_t^{2''}$ represents the reconstructed data for the second variational autoencoder in the second phase. Anomaly scores are specified in the following way:

$$s = \frac{1}{2} \left\| \hat{\boldsymbol{W}} - V_1(\hat{\boldsymbol{W}}) \right\|_F^2 + \frac{1}{2} \left\| \hat{\boldsymbol{W}} - V_2(\hat{\boldsymbol{W}}) \right\|_F^2 \tag{19}$$

where $\hat{\boldsymbol{W}}$ represents the window data at the current timestamp, and is not observed. Data points are classified to be abnormal when scores are larger than the threshold λ that we set, otherwise they are considered to be normal [33].

Algorithm 2 Testing algorithm.

Input: Test windows Dataset $\hat{\boldsymbol{W}} = \left\{ \hat{\boldsymbol{W}}_1, ..., \hat{\boldsymbol{W}}_T' \right\}$, threshold λ

Output: Labels y: $\{ y_1, ..., y_{T'} \}$

for $t = 1$ *to* T' **do**

- $\hat{\boldsymbol{W}}_t^{1'} \leftarrow D_1(E(\hat{\boldsymbol{W}}_t))$
- $\hat{\boldsymbol{W}}_t^{2''} \leftarrow D_2(E(\hat{\boldsymbol{W}}_t^{1'}))$
- $s \leftarrow \frac{1}{2} \left\| \hat{\boldsymbol{W}}_t - \hat{\boldsymbol{W}}_t^{1'} \right\|_F^2 + \frac{1}{2} \left\| \hat{\boldsymbol{W}}_t - \hat{\boldsymbol{W}}_t^{2''} \right\|_F^2$
- **if** $s \geq \lambda$ **then**
  - $y_t \leftarrow 1$
- **else**
  - $y_t \leftarrow 0$

Table 2
Benchmark Datasets. (%) represents the percent of abnormal data points.

Data set name	Sub set number	Dimension number	Training set size	Testing set size	Anomaly ratio (%)
SMD	28	38	708405	748420	4.16
MSL	57	55	58317	73729	10.72
SMAP	55	25	135183	427617	13.13
SWaT	1	51	475200	449919	10.72
PSM	1	25	132481	87841	27.8

5. Experiments and results

The algorithm is implemented, with the use of PyTorch (V2.0.0) and Python (3.8.16). A machine running the Windows 11 operating system is used for all experiments, equipped with an NVIDIA GeForce RTX 3060 Laptop GPU and 6 GB of memory. In the experiment, we basically summarize the default hyper-parameters used. Specifically, the batch size of the training, validation and testing datasets is uniformly set to 128. The weights of the network model are optimized by the Adam optimizer and the initial learning rate is set to 10^{-4}. In the training phase, the total number of epochs is executed for 250. The number of early stopping is set to 5 in order to be able to terminate the experiment when the improvement is no longer significant.

5.1. Setup

In our experiments, five datasets are used: Server Machine Dataset (SMD) [21], Soil Moisture Active Passive satellite (SMAP), Mars Science Laboratory Rover (MSL) [34], Secure Water Treatment (SWaT) [35], and Pooled Server Metrics (PSM) [36]. Detailed features of datasets are summarized in Table 2.

- SMD is a new dataset collected and publicly released by a major Internet company. The dataset covers a 5-week time span and contains data from 28 server machines, with a 1-minute time interval between each neighboring data. Each machine is monitored by 33 metrics. The SMD dataset is divided into two equal-sized subsets: the first half is used as a training set (without labels), while the second half is used as a test set (with labels).
- SMAP and MSL are the two public datasets of the real-world dataset series, both labeled by NASA experts. These datasets provide us with important information about soil moisture and Mars science experiments. The SMAP dataset contains data from 55 entities, while the MSL dataset contains data from 27 entities. Each entity is monitored using either 25 or 55 indicators.
- SWaT is an invaluable resource for the study of industrial control systems and cyber security. It provides an environment that simulates a water treatment system with various types of data and simulated attacks. The dataset is from a single entity and each entity monitors 51 metrics.
- PSM is a public dataset from eBay server machines. The dataset contains data from one entity, each monitoring 25 metrics.

In addition, we use the precision (P), recall (R), F1 score (F1), and F1* score (F1*) [37] to assess the performance of our method and baselines:

$$P = \frac{TP}{TP + FP} \tag{20}$$

$$R = \frac{TP}{TP + FN} \tag{21}$$

$$F1 = 2 \cdot \frac{P \cdot R}{P + R} \tag{22}$$

$$F1^* = 2 \cdot \frac{\bar{P} \cdot \bar{R}}{\bar{P} + \bar{R}} \tag{23}$$

where TP is the True Positives, FP is the False Positives, and FN is the False negatives. $\bar{P}$ and $\bar{R}$ represent the average precision and recall, respectively.

5.2. Overall performance

In practical production scenarios, anomalies seldom occur in isolation but rather persist for a period of time, forming continuous abnormal segments. Therefore, our method does not focus on how well a single point in time operates, but on data points over a period of time. In view of such issues, the point adjustment approach [37] is used for optimizing the model. When an anomaly can be detected within a time period, that period is an anomaly segment, and the points in it are identified as anomalies.

To validate the overall performance of our method, baselines are used for comparisons: VAE [38], LSTM [39], LSTM-VAE [40], USAD [37], and BeatGAN [29]. Since not all detection methods employ an identical threshold determination strategy, for the sake of unification, all methods compared in this section adopt the same strategy for each possible threshold in the testing phase of the model. Table 3 presents detailed performance outcomes achieved by all methods for datasets. Our method outperforms all other methods

Table 3
Performance Comparison. Precision (P), Recall (R), F1 scores and F1* scores on the public datasets.

Methods	SMD				MSL			
	P	R	F1	F1*	P	R	F1	F1*
VAE	0.8703	0.7488	0.7763	0.8050	0.8785	0.9897	0.9096	0.9308
LSTM	0.8852	0.8682	0.8596	0.8766	0.8500	**0.9999**	0.8931	0.9189
LSTM-VAE	0.8950	0.8764	0.8707	0.8856	0.8147	0.9805	0.8555	0.8899
USAD	0.8124	0.7438	0.7493	0.7766	0.8852	0.9400	0.8866	0.9118
BeatGAN	0.8817	0.8408	0.8525	0.8608	0.8951	0.9743	0.9199	0.9330
Ours	**0.9518**	**0.9654**	**0.9571**	**0.9586**	**0.9522**	**0.9999**	**0.9735**	**0.9755**
Methods	SMAP				SWaT			
	P	R	F1	F1*	P	R	F1	F1*
VAE	0.7288	0.9821	0.7730	0.8367	0.9494	0.7604	0.8444	0.8444
LSTM	0.7240	0.9922	0.7723	0.8371	0.4387	0.5426	0.5912	0.5912
LSTM-VAE	0.7468	0.9880	0.7917	0.8506	**0.9526**	0.7190	0.8195	0.8195
USAD	0.7144	0.9938	0.7678	0.8312	0.9196	0.8110	0.8619	0.8619
BeatGAN	0.7318	0.9510	0.7757	0.8271	0.5644	0.8310	0.6723	0.6723
Ours	**0.8943**	**0.9951**	**0.9296**	**0.9420**	0.9363	**0.9350**	**0.9357**	**0.9357**
Methods	PSM				Total			
	P	R	F1	F1*	P	R	F1	F1*
VAE	0.8483	0.8319	0.8400	0.8400	0.8550	0.8625	0.8286	0.8513
LSTM	0.9597	0.8971	0.8780	0.8780	0.7715	0.8600	0.7988	0.8203
LSTM-VAE	0.9209	0.9193	0.9201	0.9201	0.8660	0.8966	0.8515	0.8731
USAD	**0.9775**	0.8130	0.8877	0.8877	0.8618	0.8603	0.8306	0.8538
BeatGAN	0.9111	0.8193	0.8627	0.8627	0.7968	0.8832	0.8166	0.8311
Ours	0.9578	**0.9674**	**0.9626**	**0.9626**	**0.9384**	**0.9725**	**0.9517**	**0.9548**

across all datasets (SMAP, MSL, SMD, SWaT, and PSM), surpassing the best-performing benchmark F1 scores by 17.4%, 5.8%, 9.9%, 8.5%, and 11.7% respectively. This is because although VAE reconstructs raw data better, it does not extract the temporal features of raw data, which VAEAT does well. Although LSTM-VAE compensates for this problem, it ignores the inherent distribution of time series data and the noise in raw data, whereas VAEAT's two-phase training strategy effectively addresses these concerns. Thus, our method demonstrates substantial enhancement and excellent performance.

5.3. Visualization of anomaly scores

In order to get a clearer understanding of various methods' performance, we visually analyze their normalized anomaly scores. This operation is performed on the SMD dataset. Fig. 3a-3f represent the point plots of anomaly scores for VAE, LSTM, LSTM-VAE, USAD, BeatGAN, and VAEAT, respectively. In the above six plots, x-axis denotes the time axis, y-axis denotes normalized anomaly scores, blue dots denote anomaly scores for each point, red dotted lines represent the thresholds used for determining if each time point is abnormal or not, and the shaded area indicates that the current time point is an anomaly. As can be seen from Fig. 3a, due to the lack of extracted temporal features, the VAE model suffers from many normal points being misidentified as anomalies, leading to a decrease in the precision and recall. From Fig. 3b, it can be seen that although LSTM makes a significant increase in the precision and recall due to the extraction of temporal features, it is slightly deficient in detecting some anomalies very close with normal (anomaly scores of anomalies are very close to the threshold), i.e., it is susceptible to noise. As can be seen from Fig. 3c, LSTM-VAE still has shortcomings, and the precision and recall rate are still low. In order to better address the noise effect, it is clear from Fig. 3d and Fig. 3e in which USAD and BeatGAN perform slightly better after applying the adversarial idea, but its precision and recall rate are still low due to the lack of temporal feature extraction. As can be seen from Fig. 3f, our method can better identify the anomalous data that is very similar to the normal.

5.4. Effect of parameters

Effects of various parameters for our method are evaluated through experiments.

Window size. Experiments on this parameter are performed on the SMAP dataset. Fig. 4a illustrates the results of experiments with four different window sizes. When the window size changes from small to large, the F1 scores show a tendency to rise and then fall. As the window size gets smaller, lower F1 scores are obtained. This is because small windows do not represent the context information of window data well. As the window size gets larger, lower F1 scores are obtained. The reason for this is that the larger the window that can be observed, the harder it is to detect small segment abnormalities incorporated into the long segment data. However, the F1 scores of VAEAT are still much higher than those of baselines, indicating VAEAT still maintains high performances when dealing with long sequences.

The dimension size m of the latent variable. Experiments on this parameter are performed on the SMAP and PSM datasets. From Fig. 4b and Fig. 4c, it can be seen that as m increases, the F1 scores behave differently on the two datasets. From Fig. 4b, it can

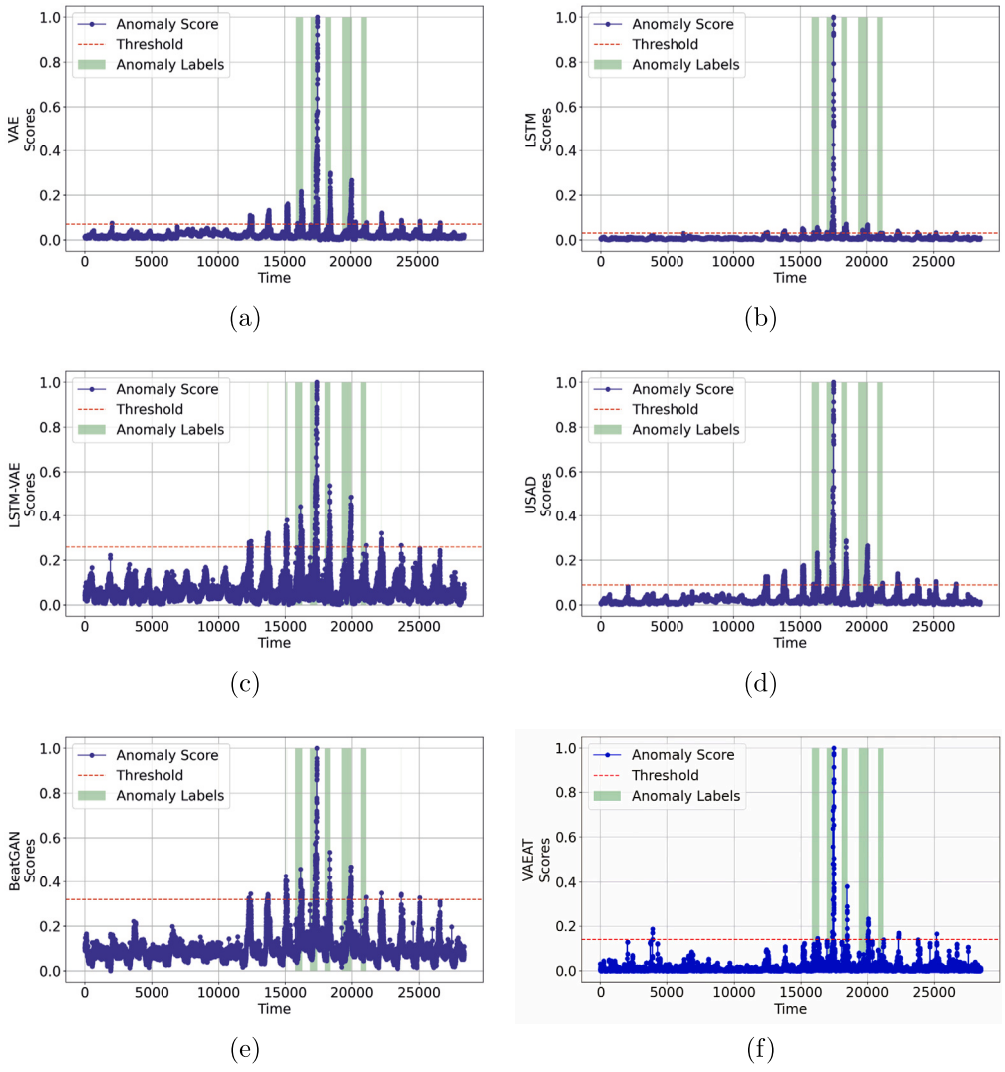


Fig. 3. The visualization of anomaly scores of different methods on the SMD dataset. 1) Blue dots: anomaly scores for each time point. 2) Red dotted lines: thresholds for determining whether each time point is an anomaly. 3) Shaded areas: current points in time are anomalies.

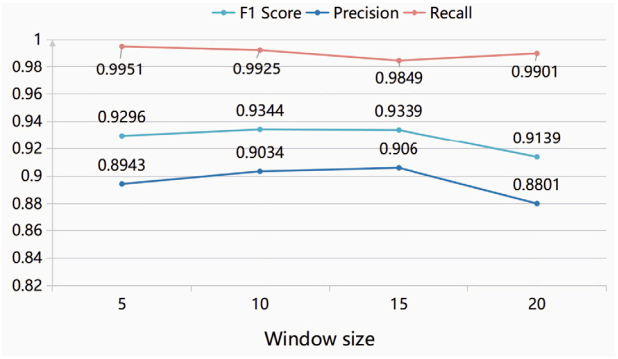
be seen that F1 scores fluctuate more with increasing m on the SMAP dataset. From Fig. 4c, it can be seen that the F1 scores remain relatively stable on the PSM dataset as m increases. We speculate that the reason for this difference is the subset size of the dataset. After the data from the SMAP dataset is reallocated to the subset, the amount of subset data may not be sufficient to adequately learn the sequence features. As a result, the redundant information in the latent variables has a more pronounced effect on the final results, leading to greater fluctuations in the F1 score curves. Whereas, the PSM dataset has enough data to learn the sequence features, and the effect of redundant information on the final results is negligible, making the F1 score curve remain stable.

5.5. Noise experiment

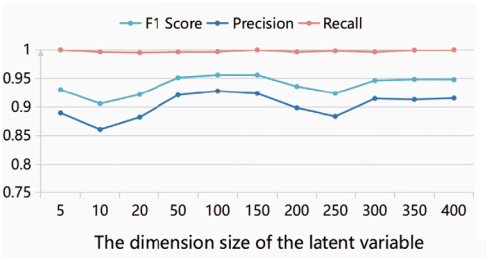
In this subsection, we examine the impact of noise on each method. We employ two datasets, MSL and SMAP, to conduct our experiments.

From results achieved on the MSL dataset, illustrated by Fig. 5a, it is observed that other methods experience a sharp decrease in F1 scores after being affected by noise, with an average drop between 4% and 9%. This indicates that they are highly affected by noise. In contrast, our method exhibits robust performance against noise on the MSL dataset, without noticeable degradation.

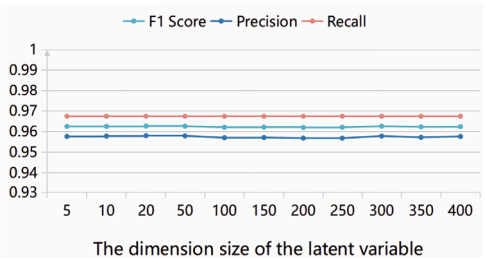
As depicted in Fig. 5b, the F1 scores of the comparison methods exhibit slight fluctuations at the initial stages of introducing noise into the SMAP dataset. As noise levels rise, their performance demonstrates an overarching decline. In contrast, our method maintains its performance well on the SMAP dataset after being affected by noise, which is almost unaffected except for a slight drop at the beginning.



(a)



(b)



(c)

Fig. 4. Effect of the window size and the dimension size of the latent variable on the precision (P), recall (R) and F1 score. (a) The values of the window size on the SMAP dataset are 5, 10, 15 and 20. Y-axis indicates values of the corresponding evaluation indexes. As the window size gets larger, the F1 score tends to decrease. (b) The values of the dimension size of the latent variable on the SMAP dataset are 5, 10, 20, 50, 100, 150, 200, 250, 300, 350 and 400. Y-axis indicates values of corresponding evaluation indexes. The F1 scores fluctuate more as the dimension size of the latent variable becomes larger. (c) The values of the dimension size of the latent variable on the PSM dataset are 5, 10, 20, 50, 100, 150, 200, 250, 300, 350 and 400. Y-axis indicates values of corresponding evaluation indexes. The F1 scores are relatively smooth as the dimension size of the latent variable becomes larger.

Fig. 5c shows the effect more clearly. The x-axis represents various methods on the MSL and SMAP datasets, and the y-axis represents the percentage decline in performance from no noise to a noise ratio of 30. The decline in the F1 score of VAEAT on the MSL dataset is very slight. Although the F1 score of our method decreases by 3.8% on the SMAP dataset, the decline is the smallest compared to other methods. In general, our method performs well under the influence of noise and shows strong resistance to noise.

5.6. Ablation study

Ablation experiments are implemented for assessing effects on anomaly detection effectiveness of critical modules in the method. We design three different variants by removing and substituting three key components. We compare our method with three variants: VAEAT-VAE, which converts the variational autoencoder into a basic autoencoder, VAEAT-LSTMA, which applies only the variational autoencoder framework, and VAEAT-A, which removes the attention mechanism from LSTM. Details are given in Table 4.

Effect of VAE: The results for the VAEAT-VAE variant show an average performance degradation of 13.7% for the three datasets when VAE is replaced by the basic AE. We argue that the latent spatial diversity has a significant influence over the model's performance.

Effect of LSTM: The results from the VAEAT-LSTMA variant show an average performance degradation of 13.6% across the three datasets when LSTM is removed. We argue that using VAE without LSTM impairs capabilities of the model for capturing temporal features of the data.

Effect of adding the attention mechanism on top of LSTM: The results for the VAEAT-A variant show that an average performance degradation of 14.7% across the three datasets when the attention mechanism attached to LSTM is removed. We argue that using LSTM without an attention mechanism impairs the model's ability to capture long sequence features.

6. Conclusions

We propose a novel unsupervised anomaly detection method for multivariate time series, named VAEAT, which uses VAEs as the main architecture and creates a two-phase training strategy using the adversarial training idea. This method not only solves the problem that VAE fails to adequately learn the underlying data distribution, but also enhances its noise resistance. Experimental results on five public datasets indicate that VAEAT is adept at detecting anomalies in time series efficiently. Compared to all baseline methods, our method demonstrates superior anomaly detection performance and increased robustness against noise. However,

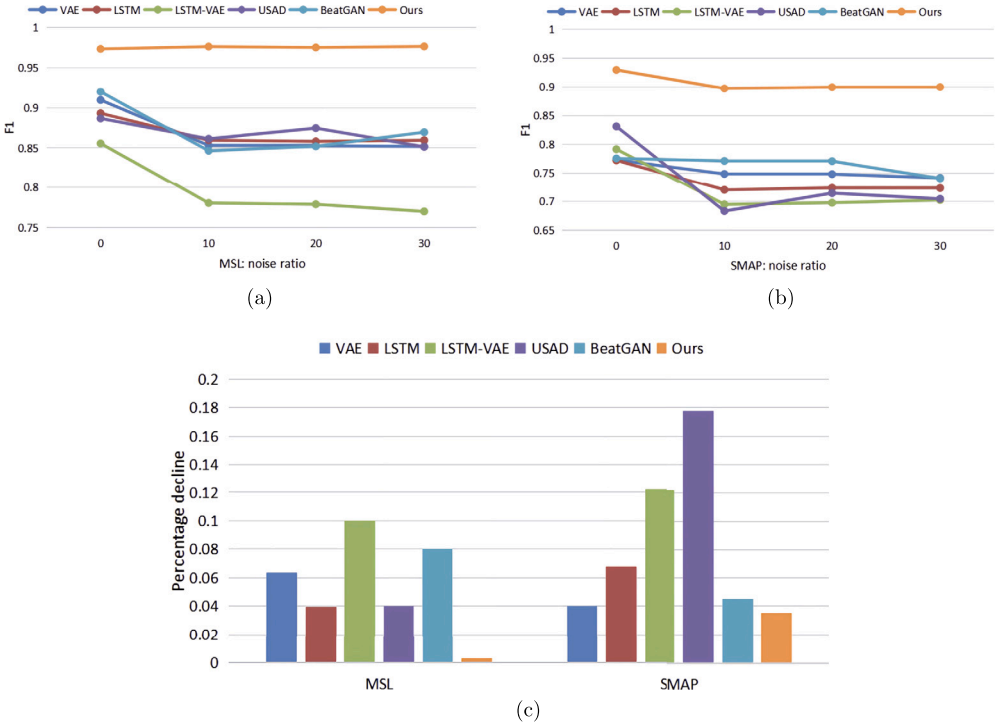


Fig. 5. The effect of noise on each method. (a) On the MSL dataset. X-axis indicates injected noise ratios of 0, 10, 20 and 30. Y-axis denotes F1 scores for methods with corresponding injected noise ratios. (b) On the SMAP dataset. X-axis indicates injected noise ratios of 0, 10, 20, and 30. Y-axis denotes F1 scores for methods with corresponding injected noise ratios. (c) X-axis represents various methods on the MSL and SMAP datasets, and y-axis denotes the percentage decline in performance from no noise to a noise ratio of 30.

Table 4
The Comparison of Variants. Marking √ indicates that this part is used.

Methods	MSL					
	LSTM	VAE	Attention	P	R	F1
VAEAT-VAE	√		√	0.8899	0.9999	0.9226
VAEAT-LSTMA		√		0.8883	0.9999	0.9232
VAEAT-A	√	√		0.8849	0.9910	0.9155
Ours	√	√	√	0.9522	0.9999	**0.9735**
Methods	SMAP					
	LSTM	VAE	Attention	P	R	F1
VAEAT-VAE	√		√	0.7292	0.9917	0.7768
VAEAT-LSTMA		√		0.7294	0.9873	0.7780
VAEAT-A	√	√		0.7520	0.9873	0.8016
Ours	√	√	√	0.8943	0.9951	**0.9296**
Methods	SMD					
	LSTM	VAE	Attention	P	R	F1
VAEAT-VAE	√		√	0.8572	0.8337	0.8236
VAEAT-LSTMA		√		0.8332	0.8245	0.8146
VAEAT-A	√	√		0.8303	0.7828	0.7838
Ours	√	√	√	0.9518	0.9654	**0.9571**

this paper does not thoroughly explore the relationship between time series attributes for detecting overall abnormalities through anomalies at a single attribute. For this reason, inter-attribute relationships will be emphasized in future work.

CRediT authorship contribution statement

Sheng He: Writing – original draft, Visualization, Software, Resources, Methodology, Formal analysis, Data curation, Conceptualization. **Mingjing Du:** Writing – original draft, Supervision, Project administration, Funding acquisition. **Xiang Jiang:** Writing –

review & editing, Visualization, Validation. **Wenbin Zhang:** Writing – review & editing, Validation. **Congyu Wang:** Writing – review & editing, Investigation.

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.

Acknowledgements

This work is supported by the National Natural Science Foundation of China (No. 62006104), Postgraduate Research & Practice Innovation Program of Jiangsu Normal University (No. 2024XKT2636).

References

[1] C. Ji, M. Du, Y. Hu, S. Liu, L. Pan, X. Zheng, Time series classification based on temporal features, Appl. Soft Comput. 128 (2022) 109494.
[2] H. Liang, L. Song, J. Du, X. Li, L. Guo, Consistent anomaly detection and localization of multivariate time series via cross-correlation graph-based encoder–decoder GAN, IEEE Trans. Instrum. Meas. 71 (2021) 3504210.
[3] L.-P. Yuan, P. Liu, S. Zhu, Recompose event sequences vs. predict next events: a novel anomaly detection approach for discrete event logs, in: Proceedings of the 16th ACM Asia Conference on Computer and Communications Security, 2021, pp. 336–348.
[4] Q. Lu, X. Xie, A.K. Parlikad, J.M. Schooling, Digital twin-enabled anomaly detection for built asset monitoring in operation and maintenance, Autom. Constr. 118 (2020) 103277.
[5] J. Cai, J. Fan, Perturbation learning based anomaly detection, Adv. Neural Inf. Process. Syst. 35 (2022) 14317–14330.
[6] H. Li, W. Zheng, F. Tang, Y. Zhu, J. Huang, Few-shot time-series anomaly detection with unsupervised domain adaptation, Inf. Sci. 649 (2023) 119610.
[7] H. Li, M. Chen, Time series clustering based on normal cloud model and complex network, Appl. Soft Comput. 148 (2023) 110876.
[8] H. Li, R. Jia, X. Wan, Time series classification based on complex network, Expert Syst. Appl. 194 (2022) 116502.
[9] H. Guo, L. Wang, X. Liu, W. Pedrycz, Trend-based granular representation of time series and its application in clustering, IEEE Trans. Cybern. 52 (9) (2021) 9101–9110.
[10] H. Li, T. Du, X. Wan, Time series clustering based on relationship network and community detection, Expert Syst. Appl. 216 (2023) 119481.
[11] Y. Bai, J. Wang, X. Zhang, X. Miao, Y. Lin, Crossfun: multiview joint cross-fusion network for time-series anomaly detection, IEEE Trans. Instrum. Meas. 72 (2023) 3532109.
[12] M. Du, Y. Wei, X. Zheng, C. Ji, Multi-feature based network for multivariate time series classification, Inf. Sci. 639 (2023) 119009.
[13] S. Kanarachos, S.-R.G. Christopoulos, A. Chroneos, M.E. Fitzpatrick, Detecting anomalies in time series data via a deep learning algorithm combining wavelets, neural networks and Hilbert transform, Expert Syst. Appl. 85 (2017) 292–304.
[14] M. Du, F. Li, G. Zheng, V. Srikumar, Deeplog: anomaly detection and diagnosis from system logs through deep learning, in: Proceedings of the 24th ACM SIGSAC Conference on Computer and Communications Security, 2017, pp. 1285–1298.
[15] W. Liao, Y. Guo, X. Chen, P. Li, A unified unsupervised Gaussian mixture variational autoencoder for high dimensional outlier detection, in: Proceedings of the 6th IEEE International Conference on Big Data, 2018, pp. 1208–1217.
[16] D. Rezende, S. Mohamed, Variational inference with normalizing flows, in: Proceedings of the 32nd International Conference on Machine Learning, 2015, pp. 1530–1538.
[17] Y. Tang, Z. Pan, X. Hu, W. Pedrycz, R. Chen, Knowledge-induced multiple kernel fuzzy clustering, IEEE Trans. Pattern Anal. Mach. Intell. 45 (12) (2023) 14838–14855.
[18] W. Wu, L. He, W. Lin, Y. Su, Y. Cui, C. Maple, S. Jarvis, Developing an unsupervised real-time anomaly detection scheme for time series with multi-seasonality, IEEE Trans. Knowl. Data Eng. 34 (9) (2020) 4147–4160.
[19] Q. Chen, A. Zhang, T. Huang, Q. He, Y. Song, Imbalanced dataset-based echo state networks for anomaly detection, Neural Comput. Appl. 32 (2020) 3685–3694.
[20] H. Zhao, Y. Wang, J. Duan, C. Huang, D. Cao, Y. Tong, B. Xu, J. Bai, J. Tong, Q. Zhang, Multivariate time-series anomaly detection via graph attention network, in: Proceedings of the 20th IEEE International Conference on Data Mining, 2020, pp. 841–850.
[21] Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun, D. Pei, Robust anomaly detection for multivariate time series through stochastic recurrent neural network, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 2828–2837.
[22] C. Zhang, D. Song, Y. Chen, X. Feng, C. Lumezanu, W. Cheng, J. Ni, B. Zong, H. Chen, N.V. Chawla, A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data, in: Proceedings of the 33rd AAAI Conference on Artificial Intelligence, 2019, pp. 1409–1416.
[23] J. Wang, S. Shao, Y. Bai, J. Deng, Y. Lin, Multiscale wavelet graph autoencoder for multivariate time-series anomaly detection, IEEE Trans. Instrum. Meas. 72 (2023) 2502911.
[24] Z. Li, W. Chen, D. Pei, Robust and unsupervised KPI anomaly detection based on conditional variational autoencoder, in: Proceedings of the 37th IEEE International Performance Computing and Communications Conference, 2018, pp. 1–9.
[25] L. Li, J. Yan, Q. Wen, Y. Jin, X. Yang, Learning robust deep state space for unsupervised anomaly detection in contaminated time-series, IEEE Trans. Knowl. Data Eng. 35 (6) (2023) 6058–6072.
[26] L. Zhang, W. Bai, X. Xie, L. Chen, P. Dong, Tmanomaly: time-series mutual adversarial networks for industrial anomaly detection, IEEE Trans. Ind. Inform. (2023).
[27] F. Kong, J. Li, B. Jiang, H. Wang, H. Song, Integrated generative model for industrial anomaly detection via bidirectional LSTM and attention mechanism, IEEE Trans. Ind. Inform. 19 (1) (2021) 541–550.
[28] H. Zenati, M. Romain, C.-S. Foo, B. Lecouat, V. Chandrasekhar, Adversarially learned anomaly detection, in: Proceedings of the 18th IEEE International Conference on Data Mining, 2018, pp. 727–736.
[29] B. Zhou, S. Liu, B. Hooi, X. Cheng, J. Ye, Beatgan: anomalous rhythm detection using adversarially generated time series, in: Proceedings of the 28th International Joint Conference on Artificial Intelligence, 2019, pp. 4433–4439.
[30] X. Chen, L. Deng, F. Huang, C. Zhang, Z. Zhang, Y. Zhao, K. Zheng, Daemon: unsupervised anomaly detection and interpretation for multivariate time series, in: Proceedings of the 37th IEEE International Conference on Data Engineering, 2021, pp. 2225–2230.

[31] P. Wesołowski, K. Walasek, W. Homenda, C. Ouyang, F. Yu, Time series classification based on fuzzy cognitive maps and multi-class decomposition with ensembling, in: Proceedings of the 32nd IEEE International Conference on Fuzzy Systems, 2023, pp. 1–8.
[32] D. Fu, Z. Zhang, J. Fan, Dense projection for anomaly detection, in: Proceedings of the 38th AAAI Conference on Artificial Intelligence, 2024, pp. 8398–8408.
[33] Y. Zhang, Y. Sun, J. Cai, J. Fan, Deep orthogonal hypersphere compression for anomaly detection, in: Proceedings of the 12th International Conference on Learning Representations, 2024, p. 7228.
[34] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, T. Soderstrom, Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 387–395.
[35] A.P. Mathur, N.O. Tippenhauer, SWaT: a water treatment testbed for research and training on ICS security, in: Proceedings of the 2nd International Workshop on Cyber-Physical Systems for Smart Water Networks, 2016, pp. 31–36.
[36] A. Abdulaal, Z. Liu, T. Lancewicki, Practical approach to asynchronous multivariate time series anomaly detection and localization, in: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, 2021, pp. 2485–2494.
[37] J. Audibert, P. Michiardi, F. Guyard, S. Marti, M.A. Zuluaga, Usad: unsupervised anomaly detection on multivariate time series, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020, pp. 3395–3404.
[38] J. An, S. Cho, Variational autoencoder based anomaly detection using reconstruction probability, Spec. Lect. IE 2 (1) (2015) 1–18.
[39] P. Malhotra, L. Vig, G. Shroff, P. Agarwal, et al., Long short term memory networks for anomaly detection in time series, in: Proceedings of the 23rd European Symposium on Artificial Neural Networks, 2015, p. 89.
[40] D. Park, Y. Hoshi, C.C. Kemp, A multimodal anomaly detector for robot-assisted feeding using an lstm-based variational autoencoder, IEEE Robot. Autom. Lett. 3 (3) (2018) 1544–1551.