Applied Intelligence (2023) 53:6074–6098
https://doi.org/10.1007/s10489-022-03829-1

Semisupervised anomaly detection of multivariate time series based on a variational autoencoder

Ningjiang Chen[1,2,3] · Huan Tu[1] · Xiaoyan Duan[1] · Liangqing Hu[1] · Chengxiang Guo[4]

Accepted: 29 May 2022 / Published online: 5 July 2022

Abstract

In a large-scale cloud environment, many key performance indicators (KPIs) of entities are monitored in real time. These multivariate time series consist of high-dimensional, high-noise, random and time-dependent data. As a common method implemented in artificial intelligence for IT operations (AIOps), time series anomaly detection has been widely studied and applied. However, the existing detection methods cannot fully consider the influence of multiple factors and cannot quickly and accurately detect anomalies in multivariate KPIs of entities. Concurrently, fine-grained root cause locations cannot be determined for detected anomalies and often require abundant normal data that are difficult to obtain for model training. To solve these problems, we propose a long short-term memory (LSTM)-based semisupervised variational autoencoder (VAE) anomaly detection strategy called LR-SemiVAE. First, LR-SemiVAE uses VAE to perform feature dimension reduction and reconstruction of multivariate time series data and judges whether the entity is abnormal by calculating the reconstruction probability score. Second, by introducing an LSTM network into the VAE encoder and decoder, the model can fully learn the time dependence of multivariate time series. Then, LR-SemiVAE predicts the data labels by introducing a classifier to reduce the dependence on the original labeled data during model training. Finally, by proposing a new evidence lower bound ($ELBO$) loss function calculation method, LR-SemiVAE pays attention to the normal pattern and ignores the abnormal pattern during training to reduce the time cost of removing random anomaly and noise data. However, due to the limitations of LSTM in learning the long-term dependence of time series data, based on LR-SemiVAE, we propose a transformer-based semisupervised VAE anomaly detection and location strategy called RT-SemiVAE for cluster systems with complex service dependencies. This method learns the long-term dependence of multivariate time series by introducing a parallel multihead attention mechanism transformer, while LSTM is used to capture short-term dependence, and the introduction of parallel computing also markedly reduces model training time. After RT-SemiVAE detects entity anomalies, it traces the root entities according to the obtained service dependence graph and locates the root causes at the indicator level. We verify the strategies by using public data sets and constructing a system prototype. Experimental results show that compared with existing baseline methods, the LR-SemiVAE and RT-SemiVAE strategies can detect anomalies more quickly and accurately and perform fine-grained and accurate localization of the root causes of anomalies.

Keywords Multivariate time series · Anomaly detection · Semisupervised learning · VAE · LSTM · Attention mechanism

1 Introduction

With the maturity and large-scale commercialization of cloud computing, software deployment and service change are more frequent, resulting in frequent cloud faults in cluster systems. If timely and effective fault management measures are not taken, massive losses may occur. For example, it was estimated that in 2016, Amazon would have lost 15 M$ for one hour of downtime [1]. Therefore, to ensure the high availability and failure elasticity of cloud service systems, the concept of reliability-driven artificial intelligence for IT operations (AIOps) has been widely studied and applied.

AIOps methods for cloud faults primarily include anomaly detection, fault diagnosis and failure prediction. As the most common method, anomaly detection can primarily be divided into anomaly detection based on log data and anomaly detection based on key performance indicators (KPIs) [2]. Log anomaly detection typically includes four

✉ Ningjiang Chen
chnj@gxu.edu.cn

Extended author information available on the last page of the article.

processes: log parsing [3], log partition, feature extraction and model training [4]. Anomaly detection based on KPIs achieves efficient online anomaly detection by monitoring runtime information of cluster systems (central processing unit (CPU) usage, network traffic, response delay, etc.), capturing dependencies of multivariate KPIs and understanding health status of entities in the cloud [5].

In recent years, recurrent neural networks (RNNs), variational autoencoders (VAEs) [6] and generative adversarial networks (GANs) [7] have been used for anomaly detection of multivariate KPIs and have achieved good results. RNNs (such as long short-term memory (LSTM) [8], gated recurrent units (GRUs), etc.) are often used as the base model of VAEs (i.e., the encoder and decoder) and GANs (i.e., the generator and discriminator) to capture the time dependence of multivariate KPIs. VAEs and GANs can simultaneously learn the dependencies of multivariate KPIs in the feature dimension and the complex distribution of KPIs in the temporal dimension. For example, Li et al. [9] used LSTM as the base model in the GAN framework to capture the temporal correlation of time series distributions and proposed the multivariate anomaly detection (MAD)-GAN method. Park et al. [10] used an LSTM network to replace the feedforward neural network in the VAE codec for model training and anomaly detection, proposing the LSTM-VAE algorithm. Lin et al. [11] also proposed a time series anomaly detection method based on a mixed VAE and LSTM model. Niu et al. [12] studied a mixed LSTM, VAE and GAN model and proposed a VAE-GAN time series anomaly detection method based on LSTM by jointly training the encoder, generator and discriminator. Although the application of RNNs, VAEs and GANs in anomaly detection of multivariate KPIs has achieved great performance improvement, three problems remain:

(1) Entity KPIs are multivariate in the feature dimension and are characterized by high-dimensional [13], high-noise, random and time-dependent data [14] in the temporal dimension. Existing detection methods cannot fully consider the influence of multiple factors and cannot quickly and accurately detect anomalies in multivariate KPIs of entities.

(2) Existing methods must screen out many normal data for model training, while the occurrence of anomalies in complex cloud environments is random, and multivariate KPIs often have noise data. For example, when the cloud services perform normal expansion or scaling operations, their KPIs will change suddenly, so it is necessary to annotate the KPI data to avoid misjudgment. This process will consume a lot of time, which does not meet the real-time requirements of online detection of cluster systems.

(3) RNNs have limitations in learning the long-term dependence of time series data and exhibit poor parallel computing ability, resulting in long model training time, and existing methods cannot locate the fine-grained root cause of the detected anomalies.

To address these three problems, the primary contributions of this paper are as follows:

- An LSTM-based semisupervised VAE (LR-SemiVAE) anomaly detection strategy is proposed. First, this strategy uses the VAE to perform feature dimension reduction and reconstruction of multivariate KPIs and judges whether the entity is abnormal by calculating its reconstruction probability score. Second, by introducing an LSTM network into the VAE encoder and decoder, the model can fully learn the time dependence of multivariate KPIs. Finally, LR-SemiVAE predicts the data labels by introducing a classifier to reduce the dependence on the original labeled data during model training.
- A new evidence lower bound ($ELBO$) loss function calculation method is proposed, which enables LR-SemiVAE to focus on the normal pattern and ignore the abnormal pattern during training to reduce the cost of removing random anomalies and noise data. Furthermore, it is unnecessary to sacrifice the time dependence of multivariate time series to screen out all normal data fragments.
- Based on LR-SemiVAE, we propose an anomaly detection and location strategy called RT-SemiVAE for cluster systems with complex dependencies. This strategy learns the long-term dependence of multivariate time series by introducing a parallel multihead attention mechanism transformer, while the LSTM network is used to capture short-term dependence, and parallel computing also markedly reduces model training time. After detecting entity anomalies, RT-SemiVAE traces the root cause entities according to the obtained service dependence graph and locates the anomaly causes at the indicator level.

We verify the LR-SemiVAE and RT-SemiVAE strategies by using public data sets and constructing the system prototype multivariate anomaly detection for time series data with VAE (MAD-VAE). Experimental results show that the anomaly detection performances of the proposed strategies are approximately 4%-50% higher than those of the three baseline algorithms with the best existing performance. The proposed strategies can detect anomalies more quickly and accurately and perform fine-grained and accurate localization of the root causes of anomalies.

The remainder of this paper is organized as follows. Section 2 introduces related work and summarizes the

research status of anomaly detection and the existing problems and shortcomings. Section 3 introduces the proposed LR-SemiVAE and RT-SemiVAE strategies in detail and explains their design ideas and algorithm process. In Section 4, the LR-SemiVAE and RT-SemiVAE strategies are experimentally verified at the algorithm and practical application levels, respectively. Then, Section 5 concludes the paper.

2 Related work

In general, anomaly detection of cluster systems for AIOps primarily has two research topics: anomaly detection based on log events and anomaly detection based on entity KPIs [2]. Log anomaly detection achieves anomaly detection by performing feature engineering on the parsed and preprocessed system log data [4], and KPI anomaly detection achieves online anomaly detection by describing the health state of the performance indicators at the cluster operation [5]. However, compared with the univariate KPI anomaly detection algorithms, the entity-oriented multivariate KPI anomaly detection algorithms consider the correlation between different KPIs of the entity (e.g., heterogeneous KPIs of the same entity tend to show similar fluctuation trends), which can learn more complex knowledge; thus, multivariate time series anomaly detection has higher intuitiveness, accuracy and detection efficiency [8], which is more conducive to saving operation and maintenance costs in large-scale cluster systems.

In the anomaly detection method based on KPIs, RNNs, VAEs [6] and GANs [7] have been widely used. RNNs (e.g., LSTM, GRU networks) can process time series data and capture the time dependence of entity KPI data. RNNs are often used for anomaly detection of multivariate time series combined with VAEs and GANs. As two common generative models, the VAE and GAN emphasize learning the rules or distribution of data generation; thus, to better characterize and model the data, the implicit feature representation of multivariate time series data must be described. Both methods use random noise when generating data and must measure the distribution difference between noise and training data when modeling the data distribution, but the modeling principles and training methods of the two are different.

The survey in [15] analyzed multivariate (or univariate) time series anomaly detection based on an LSTM and divided the time series anomaly detection into LSTM-based approaches, encoder-decoder-based approaches and hybrid model approaches. Ergen et al. [16] used LSTM and GRU networks as the feature extractors of time series data, used the one-class support vector machine (OC-SVM) and support vector data description (SVDD) algorithms as anomaly detectors, and developed an unsupervised anomaly detection algorithm that can process variable-length sequences. To address the imbalance and high-dimensional problems in time series data, Zhou et al. [17] used the encoder-decoder scheme based on LSTM to perform feature dimension reduction and reconstruction of time series and proposed a variational LSTM (VLSTM) anomaly detection method for industrial KPI data. However, Huang et al. [18] has shown that the random variable model with an appropriate probability distribution (e.g., VAE) can capture the randomness of multivariate time series data better than the deterministic variable model.

Regarding anomaly detection with LSTM and VAE (or autoencoder (AE)) hybrid models, Park et al. [10] proposed a multimodal (the fusion of high-dimensional and heterogeneous modalities) anomaly detector LSTM-VAE based on an LSTM network and VAE. This detector fuses the fused time series signal, reconstructs its expected distribution by introducing a progress-based varying prior, and then determines whether the entity is abnormal according to the reconstruction-based anomaly score and state-based threshold. Lin et al. [19] used VAE to form strong local features on short windows and used LSTM to estimate the long-term correlation of sequences based on the features deduced from the VAE module. Then, a VAE-LSTM hybrid model is proposed as an unsupervised method for time series anomaly detection. Sepehr et al. [20] introduced a probability criterion based on the classical central limit theorem to label data dynamically and proposed an unsupervised anomaly detection method based on the LSTM-VE hybrid model. These methods based on LSTM and VAE (or AE) hybrid models require a large amount of normal data to train the model to ensure good accuracy. In a complex and volatile cloud environment, the occurrence of anomalies is often random and accompanied by noise data. Therefore, determining how to select normal data is a problem that must be solved.

Regarding anomaly detection with LSTM and GAN hybrid models, the unsupervised time series data anomaly detection methods MAD-GAN [9], time series anomaly detection GAN (TadGAN) [21] and time series anomaly detection with GAN (TAno-GAN) [22] all use LSTM as the base model (i.e., the generator and discriminator) in the GAN framework to capture the temporal correlation of time series distributions and improve the anomaly scoring mechanism by proposing new reconstruction error calculation methods, thus improving the accuracy of anomaly detection.

However, time series anomaly detection methods based on GANs must find the best mapping from real-time space to the latent space at the anomaly detection stage, which produces new errors and requires long computation times. To solve this problem, Niu et al. [12] studied a

mixed model that contains LSTM, VAE and GAN elements and proposed a VAE-GAN time series anomaly detection method based on LSTM by jointly training the encoder, generator and discriminator in the VAE and GAN models to take advantage of the mapping ability of the encoder and the discrimination ability of the discriminator.

The LSTM network is often used to model the dependency of time series, but the efficiency and performance of modeling long-sequence dependencies cannot meet the requirements of large-scale clouds, while the transformer [23] allows information to flow at any length, which can better overcome the shortcomings of the LSTM model. To learn the position information of sequence data, the position embedding technique is introduced in the transformer. However, because it treats all the positions of the sequence equally, some position information is lost, resulting in a weak local position signal [24, 25]. Thus, Zhou et al. [26] used an RNN to capture the local structure of the time series, implemented a transformer to learn the long-term dependence, and proposed the complementary model R-transformer between an RNN and transformer.

Thus, existing studies cannot fully consider the influence of multiple factors on multivariate KPIs anomaly detection. Supervised learning algorithms require a lot of labeled data for training and can only detect known types of anomalies. Unsupervised learning methods must screen out many normal data for model training or have low accuracy and often make assumptions that are not consistent with reality. Therefore, existing anomaly detection methods cannot achieve multiobjective optimization in the required labeled data amount, efficiency, real-time performance and accuracy of anomaly detection, and fine-grained root cause localization cannot be performed on the detected anomalies. Therefore, these methods are not suitable for complex and volatile cluster systems. However, Bian et al. [27] designed a semisupervised anomaly detection method using a small amount of labeled data to drive many unlabeled data to train together and achieved good performance. He et al., in a survey [3], showed that compared with traditional machine learning methods (e.g., principal component analysis (PCA), clustering, SVM, frequent pattern mining, and graph mining), deep learning methods provide marked advantages in online detection (e.g., LSTM-based models) and can thus better meet the performance requirements of anomaly detection.

3 Anomaly detection and location of multivariate time series based on a semisupervised VAE

In this section, we propose LR-SemiVAE and RT-SemiVAE strategies for multivariate time series anomaly detection and fine-grained root cause localization oriented to AIOps in cloud environments. We thus introduce the framework, network structure, training process and online detection process of LR-SemiVAE in **3.1** and describe the overall architecture, model structure, anomaly detection and positioning process of RT-SemiVAE in **3.2**. The LR-SemiVAE anomaly detection strategy can capture the local dependence of multivariate time series, primarily for a single application or service. The RT-SemiVAE anomaly detection and location strategy is optimized based on LR-SemiVAE. This strategy can learn the local and long-term dependencies of time series data concurrently, primarily for application systems with complex service dependencies and invocation relations.

3.1 Related theories and definitions

The VAE is a deep Bayesian network that mainly includes an encoder and decoder. Its purpose is to reconstruct the input data x so that the reconstructed $\overline{x}$ is as close as possible to the input x. The encoder maps the input data x into latent space by $f(W_z x + b_z)$ to obtain a new feature z, and then the decoder maps the feature z from the latent space to the input space by $f(W_x z + b_x)$ to reconstruct $\overline{x}$, that is, $W_x = W_z^T$. The VAE is commonly used in feature dimension reduction tasks, often maximizing $ELBO$ as a model training target, and the performance is better than other dimension reduction techniques (such as PCA [3] et al.).

The VAE uses the prior distribution $p_\theta(z)$ to restrict the potential encoding z to a random variable distribution, which makes it easier to learn the complex distribution of high-dimensional data. It uses the feedforward neural network as the encoder and decoder. The encoder $p_\theta(z|x)$ encodes the visible variable x into the potential variable z and then reconstructs $\overline{x}$ with the same dimension as x by the decoder $p_\theta(x|z)$. Since the real posterior distribution $p_\theta(z|x)$ of continuous potential variable z is difficult to solve, VAE obtains a certain approximate posterior $q_\phi(z|x)$ by variational inference technology, where $p_\theta(x|z)$ and $q_\phi(z|x)$ are feedforward neural networks with a θ parameter and a ϕ parameter, respectively. Thus, the VAE encoding and decoding process can be expressed as $z \leftarrow Encoder(x) = q_\phi(z|x)$, $\overline{x} \leftarrow Encoder(z) = p_\theta(x|z)$; in the VAE architecture, the entire generation model can be expressed as $p_\theta(x, z) = p_\theta(x|z)p_\theta(z)$.

However, the feedforward neural network of the original VAE assumes that the data are independent at each time point, and the output of the network depends only on the current input. However, an important time dependence exists between the inputs of time series data. Therefore, it is necessary to introduce the RNN and transformer into the encoder and decoder of the VAE to learn the time dependence of multivariate KPIs.

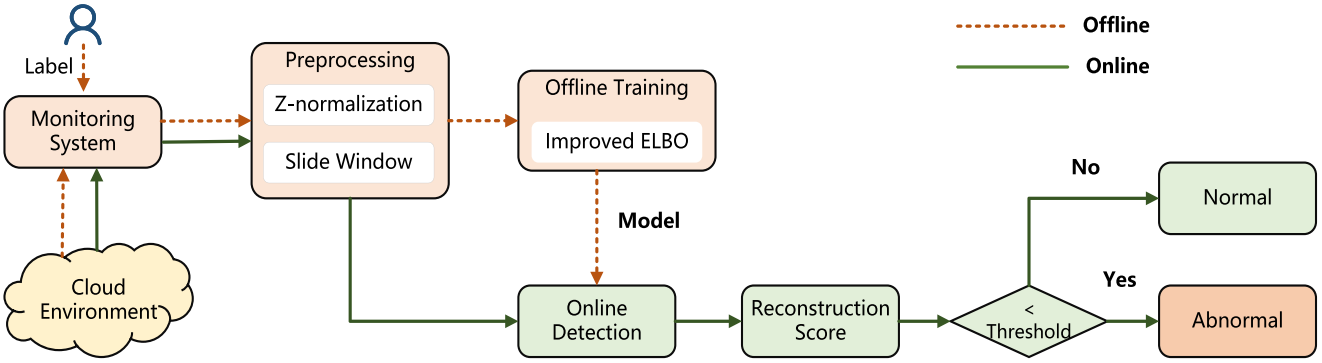


Fig. 1 LR-SemiVAE framework

The following definitions are given:

Definition 1 Multivariate time series, also known as multivariate KPIs in this paper, are multiple performance indicators collected by the cluster monitoring system according to a certain time interval. $x = (x^{(1)}, x^{(2)}, \cdots, x^{(N)})$, where N is the monitoring time of the multivariate time series x, and $x^{(t)} \in R^{d_M}$ is a M-dimensional vector, which represents the monitored indicator value at time t; $x^{(t)} = [x_1^{(t)}, x_2^{(t)}, \cdots, x_M^{(t)}], x \in R^{d_{M*N}}$.

Definition 2 Multivariate time series anomaly detection for any time t calculates the reconstruction probability score of $x^{(t)}$ through historical sequence fragment $x^{(t-w+1:t)}$=$(x^{(t-w+1)}, x^{(t-w+2)}, \cdots, x^{(t)})$ instead of directly using the value at time t, where $x^{(t-w+1:t)} \in R^{d_{M*w}}$, and then determines whether $x^{(t)}$ is abnormal according to the set threshold: if the reconstruction probability score is less than the threshold, then it is judged to be an abnormal point; otherwise, it is a normal point.

3.2 LR-SemiVAE anomaly detection strategy

3.2.1 Framework of LR-SemiVAE

As shown in Fig. 1, the LR-SemiVAE anomaly detection framework is divided into offline training and online detection, and the process is distinguished by dotted lines and solid lines. Operation and maintenance engineers can mark some data collected by the monitoring system according to their domain knowledge and then drive most unlabeled data to train together in the offline anomaly detection model. The preprocessing module is a common module of both offline and online parts, including Z-normalization and slide window; thus, each indicator conforms to the standard normal distribution, and the sliding window is used to divide x into subsequences with a moving step of one unit. To better determine the optimal window length of the sequence, different window sizes are used to determine the entity health state under different lengths. The label of the window can be determined by whether there are abnormal data in the window, and the entire window is marked as abnormal with abnormal data; otherwise, it is normal, in which the length of the historical subsequence fragment (w) is adjusted according to the data time interval. The reconstruction probability score of $x^{(t)}$ at time t is calculated using the multivariate time series fragment $x^{(t-w+1:t)}$, and then, the preprocessed $N - w$ multivariate subtime series $(x^{(t-w+1:t)}, y_l^{(t-w+1:t)})$, where $y_l^{(t-w+1:t)}$ is the label at time t, 0 indicates "normal", and 1 indicates "anomaly", are used to train together in the offline model. In addition, to improve the reconstruction performance of the model, the $ELBO$ loss function is modified to make the model focus on normal data and ignore abnormal data as much as possible.

The online module preprocesses the new data collected by monitoring, calculates the abnormal scores of all points (e.g., the score at time t is $p(y_l^{(t)} = 1|x^{(t-w+1:t)})$, where y_l is the abnormal result) according to the offline trained model, and then judges according to the set threshold. If the reconstruction score at time t is less than the threshold, an entity is judged as abnormal; otherwise, it is normal.

LR-SemiVAE first uses LSTM to capture the complex time dependence of multivariate time series and then uses the VAE to map the input multivariate subtime series $(x^{(t-w+1:t)}, y_l^{(t-w+1:t)})$, $t \in \{1, 2, \cdots, N\}$ to the random variable z space through the dimension reduction of the $q_\phi(z|x)$ encoder (LSTM-Encoder). The distribution $p_\theta(z)$ on z is constrained to the multivariate normal distribution $N(0, I)$, and the prior distribution $q_\phi(z|x)$ of z is the diagonal Gaussian distribution $N(\mu, \sigma^2, I)$, where μ is the mean and σ^2 is the variance. LR-SemiVAE uses a

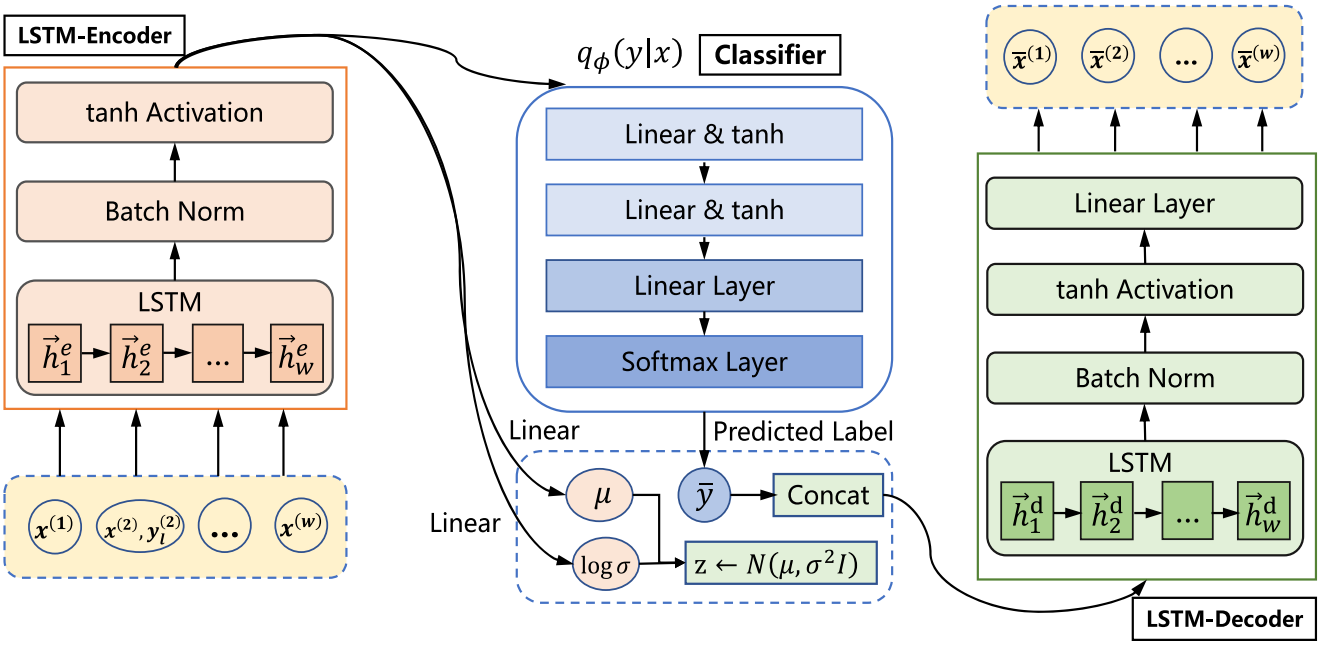


Fig. 2 Network structure of LR-SemiVAE

semisupervised learning method, which uses a small amount of labeled data to drive most of the unlabeled data to train the classifier. Then, the predicted labels $\overline{y}$ and z are used for $p_\theta(x|(z,\overline{y}))$ decoding (LSTM-Decoder), and the reconstructed input sequence is expressed as $\overline{x}^{(t-w+1:t)}$. Figure 2 shows the entire encoding and decoding process of LR-SemiVAE using an example of a subsequence fragment $(x^{(1:w)}, y_l^{(1:w)})$.

3.2.2 Training and anomaly detection process

In practical training, a joint approximate posterior can be defined as $q_\phi(y,z|x) = q_\phi(z|x)\cdot q_\phi(y|x)$, and LR-SemiVAE must consider two cases to optimize training the objective function. In the first case, for labeled data, the improved $ELBO$ loss function is shown in (1), where $a_t = 0, t \in \{1,2,\cdots,w\}$ represents that the $x^{(t)}$ at time t is abnormal; otherwise, $a_t = 1$. $\kappa = (\sum_{t=1}^{w} a_t)/w$ are the proportions of normal points in x. When the abnormal points are exposed, the role of $p_\theta(x^{(t)}|y,z)$ can be directly excluded by a_t, and the contributions of $p_\theta(z)$ and $p_\theta(y)$ can be calculated by the product of κ, while $q_\phi(z|x,y)$ is only the mapping from (x,y) to z, without considering whether it is a normal data point; thus, it is not necessary to modify:

$$\begin{aligned}\zeta_{ELBO}^{L} = E_{q_\phi(z|x,y)}[&\sum_{t=1}^{w} a_t \log p_\theta(x^{(t)}|y,z) \\ &+ \kappa \log p_\theta(y) + \kappa \log p_\theta(z) - \log q_\phi(z|x,y)] \quad (1)\end{aligned}$$

For unlabeled input data, the above interference method of reducing outliers is still available, and the variational lower bound of unlabeled data can be expressed by (2):

$$\begin{aligned}\zeta_{ELBO}^{U} = E_{q_\phi(y,z|x)}[&\sum_{t=1}^{w} a_t \log p_\theta(x^{(t)}|y,z) \\ &+\kappa \log p_\theta(y) + \kappa \log p_\theta(z) - \log q_\phi(y,z|x)] \quad (2)\end{aligned}$$

The $ELBO$ that satisfies both of the above situations can be expressed as (3):

$$\zeta_{ELBO} = \sum_{(x,y)\in L} \zeta_{ELBO}^{L}(x,y) + \sum_{x\in U} \zeta_{ELBO}^{U}(x) \quad (3)$$

In $ELBO$, the label prediction distribution $q_\phi(y|x)$ is only related to the unlabeled ζ_{ELBO}^{U}. To enable the classifier to learn with labels, a classification loss is added to the objective function. The extended $ELBO$ is (4), where the hyperparameter λ is used to balance the use of direct labeled data and predicted labeled data. Using this objective function, the data with and without labels can be correctly evaluated. Finally, the gradient descent method is used to update the parameters in the coding network and the decoding network:

$$\begin{aligned}\zeta_{ELBO}^{new} = &\sum_{(x,y)\in L} \zeta_{ELBO}^{L}(x,y) + \sum_{x\in U} \zeta_{ELBO}^{U}(x) \\ &+\lambda \sum_{(x,y)\in L} \log q_\phi(y|x) \quad (4)\end{aligned}$$

Algorithm 1 LR-SemiVAE training algorithm.

Input: training data (x, y_l), length of sliding window w.
Output: classifier parameters $Classifier$, coding network parameters ϕ, decoding network parameters θ.

$Classifier, \theta, \phi \leftarrow$ Initialize network parameters;
// Initialization parameter.
$(x, y_l) \leftarrow$ Preprocessing (x, y_l, w);
// Pretreatment data.
repeat
  $h_f^e \leftarrow$ LSTM-Encoder(x);
  // Encoding x using LSTM to obtain hidden variable.
  $\mu \leftarrow W_u^T f_\phi(h_f^e) + b_u, \log\sigma \leftarrow W_\sigma^T f_\phi(h_f^e) + b_\sigma$;
  /* If the input data point has a label, the value is assigned directly; otherwise, the classifier is used to predict the label. */
  if y_l *is* $NULL$ **then**
    $\overline{y} \leftarrow y_l$;
  else
    $\overline{y} \leftarrow Classifier(h_f^e)$;
  $z \leftarrow Sample(u, \log\sigma)$;
  $\overline{x} \leftarrow$LSTM-Decoder$(Concat(z, \overline{y}))$;
  // Mosaic and decode predicted labels and potential variables
  $Classifier, \theta, \phi \leftarrow$ Update parameters using gradients;
  // Update parameters.
until ζ *maximization and convergence*;
return $Classifier, \theta, \phi$;

Algorithm 1 describes the training process for LR-SemiVAE. The input training data $(x; y_l)$ can contain abnormal data. If y_l is *NULL*, it is expressed as unlabeled data, and the optimal ϕ, θ, and *Classifier* parameters are output.

The offline trained model can be used to determine whether the monitoring value at a given time (e.g., $x^{(t)}$ at t time) is normal. LR-SemiVAE uses the multivariate subtime series with length w as the input data (i.e., input $x^{(t-w+1:t)}$ to reconstruct $x^{(t)}$). Because $\overline{x}^{(t)}$ is reconstructed on the distribution parameters μ, σ of $x^{(t-w+1:t)}$ rather than the window itself, the probability can be used to represent the abnormal score. Thus, LR-SemiVAE uses reconstruction probability $E_{q_\phi(z|x)}[\log p_\theta(x|z)]$ as the anomaly detector and then uses the Monte Carlo method [14] to approximate the solution, which can be expressed by (5):

$$E_{q_\phi(z|x)}[\log p_\theta(x|z)] \approx \frac{1}{L}\sum_{l=1}^{L}\log(p_\theta(x|z^l)) \tag{5}$$

where L is the number of samples and $z^{(l)}(l = 1, 2, 3 \cdot \cdot \cdot L)$ is the sampling from $q_\phi(z|x)$, and then the repeated Monte Carlo gradient estimation method is used for random gradient descent. Because the reconstruction probability is negative, $Sigmoid$ function is used to transform it to the range of [0,1]. Thus, the reconstruction score $r^{(t)}$ at t can be expressed as $f(\frac{1}{L}\sum_{l=1}^{L}\log(p^\theta(x^{(t)}|z^l_{(t-w+1:t)})))$, where $f(x) = 1/(1+e^{-x})$. The higher $r^{(t)}$ is, the better the reconstruction, and the more likely $x^{(t)}$ is to be judged as normal.

The entire process of the LR-SemiVAE algorithm to obtain the reconstruction probability scores is shown in Algorithm 2. First, the multivariate time series obtained by monitoring are preprocessed to obtain multiple subsequence fragments with length w and shape $x^{(t-w+1:t)}$. Then, according to this fragment, the reconstruction probability of the last point $x^{(t)}$ is calculated, and the $Sigmoid$ function is used for conversion into the reconstruction score. For each data point $x^{(t)}$ of the multivariate time series, there is a corresponding abnormal score as the output. Finally, when the LR-SemiVAE model calculates the reconstruction probability score of the detection sequence, the state of the entity is determined according to the set threshold. If $r^{(t)}$ is higher than the set threshold, it is judged as normal and expressed as 0; otherwise, it is judged as abnormal and expressed as 1.

Algorithm 2 LR-SemiVAE algorithm for obtaining the reconstruction probability scores.

Input: Online detection data x_{test}), length of sliding window w.
Output: Reconstruction probability score r.

$x \leftarrow$ obtain multivariate time series data from x_{test};
// Read data online.
$x \leftarrow$ Preprocessing (x, w);
for $t \leftarrow w$ *to* $length(x)$ **do**
  $h_f^e \leftarrow$ LSTM-Encoder$(x^{(t-w+1:t)})$;
  $\mu \leftarrow W_u^T f_\phi(h_f^e) + b_u, \log\sigma \leftarrow W_\sigma^T f_\phi(h_f^e) + b_\sigma$;
  for $l \leftarrow 1$ *to* L **do**
    $z^l \leftarrow N(\mu, \sigma^2, I)$;
    $(\mu_{\overline{x}^l}, \sigma_{\overline{x}^l}) \leftarrow$ LSTM-Decoder$(concat(z^l, \overline{y}))$;
    $reconstructionProbability^l \leftarrow p_\theta(x|\mu_{\overline{x}^l}, \sigma_{\overline{x}^l})$;
  $rp^{(t)} \leftarrow \frac{1}{L}\sum_{l=1}^{L} reconstructionProbability^l$;
  // Reconstruction probability at time t
  $r^t \leftarrow 1/(1+e^{-rp^{(t)}})$;
  // Convert the reconstruction probability score at time t to interval [0,1] by the Sigmoid function.
return r;

In summary, LR-SemiVAE is a semisupervised recurrent neural network anomaly detection algorithm for multivariate time series. The algorithm uses LSTM to model the time dependence of data and is then used for the encoding and decoding process of the VAE. A small amount of labeled data is used to drive a large amount of unlabeled data for offline training. By improving the $ELBO$ loss function, the model pays attention to normal data during training and ignores abnormal data as much as possible; when anomalies occur, the reconstruction probability of the model will be low, making it easier to detect anomalies. In this process, it is unnecessary to sacrifice the time dependence of multivariate time series to filter out all normal data fragments, which reduces the tedious data processing and is more suitable for real environment.

3.3 RT-SemiVAE anomaly detection and location strategy

3.3.1 Framework of RT-SemiVAE

As shown in Fig. 3, RT-SemiVAE is primarily divided into offline and online parts. Specifically, the offline part includes real-time monitoring and collecting KPI data of entities (e.g., CPU utilization, memory usage, network transmission speed, etc.), labeling part of the data, analyzing and preprocessing the data, training the anomaly detection model, obtaining the dependencies between services and constructing the service dependency graph. RT-SemiVAE trains the model and updates the service dependency graph according to the set time interval. Thus, it can capture the change characteristics of multivariate KPIs in a timely manner and update the dependencies between services to improve the accuracy of anomaly detection and location. The online part includes using the trained model for anomaly detection and location, calculating the reconstruction probability score of multivariate time series, judging whether an entity is abnormal according to the set threshold (when the reconstruction probability score is less than the threshold, the entity is judged to be abnormal, and vice versa), tracing the root cause entities according to the service call chains in the service dependency graph, and locating the cause of the anomaly to the indicator level.

RT-SemiVAE obtains the dependencies between services through the service registration mechanism. When a service arrives at the service registration center, the service address and port to be invoked are not clear. Therefore, it is necessary to query the location of the service it must access and return it to the service registration center, and then, the service directly communicates with the service it must invoke. Therefore, the service registration center can obtain all service dependencies, as shown in Fig. 4. The construction of a service dependency graph is primarily divided into service registration, obtaining dependencies, saving or updating, and automatically generating a service dependency graph.

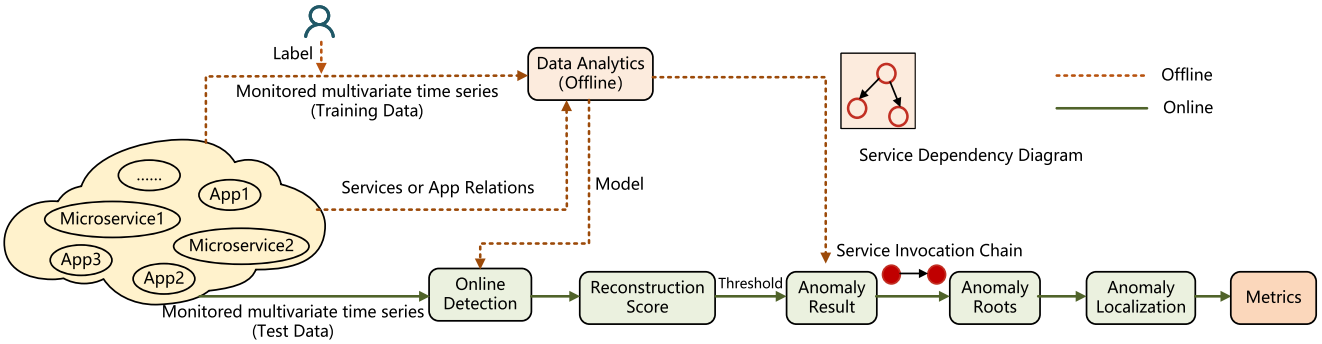


Fig. 3 RT-SemiVAE framework

Step 1: Service registration. A service registration framework is used in all subservices and applications in the cloud environment. When an application is started, all subservices send registration messages to the service registration center through the representational state transfer application programming interface (RESTful API) of the service registration framework, such as the service name, internet protocol (IP) address and communication port. Then, each service sends its own running information in real time.

Step 2: Obtaining service dependencies. Since a service needs to request the IP address and port of the target service from the service registry when it calls other services for the first time, the collection module will periodically request the service registry to obtain all service instances in the current cloud environment to establish service dependencies.

Step 3: Updating service dependencies. After the collecting module obtains all the service dependencies, RT-SemiVAE uses the graph database to save the dependencies, and the service registration center can perceive the changes in the service instances in real time and update the relevant dependencies in time. Then, the collecting module obtains the relevant information by accessing the service registration center and updates them to the graph database to obtain the service dependency graph in real time.

Step 4: Automatic generation process of the service dependency graph. A service dependency graph can be generated automatically by calling the interface of the graph database and can be visualized by the graph data parsing framework.

Due to the high dimensionality, randomness and time dependence of multivariate time series, RT-SemiVAE uses a denoising VAE to add noise $n \leftarrow N(0, \sigma^2, I)$ during coding, forcing the distribution $p_\theta(z)$ on the generated random variable z to follow the multivariate normal distribution $N(0, I)$, and the decoder reconstructs the normal $\overline{x}$ from the hidden vector z. Then, the difference between the data point of the input sequence x and the corresponding point of the reconstructed sequence $\overline{x}$ is compared, and the larger the difference is, the more likely it is to be judged as an abnormal point; otherwise, it is regarded as a normal point.

To better model the short-term and long-term dependencies of multivariate time series and to reduce the training time of the model by introducing parallel computing, the idea of a recurrent neural network enhanced transformer (R-Transformer) [26] is introduced into the VAE framework, and the location embedding technology that will cause a

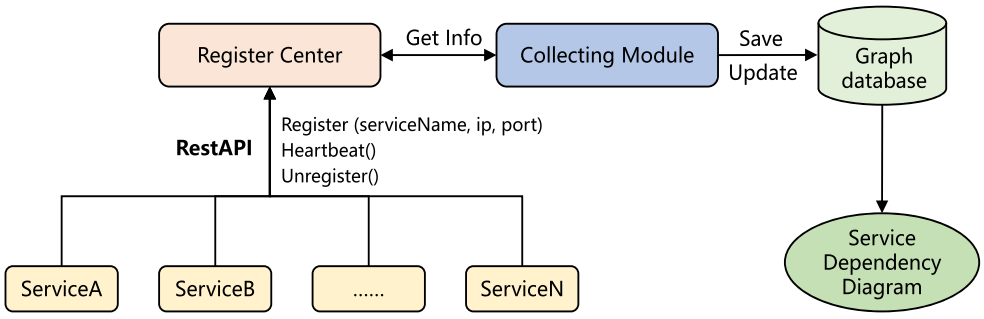


Fig. 4 Construction process of the service dependency graph

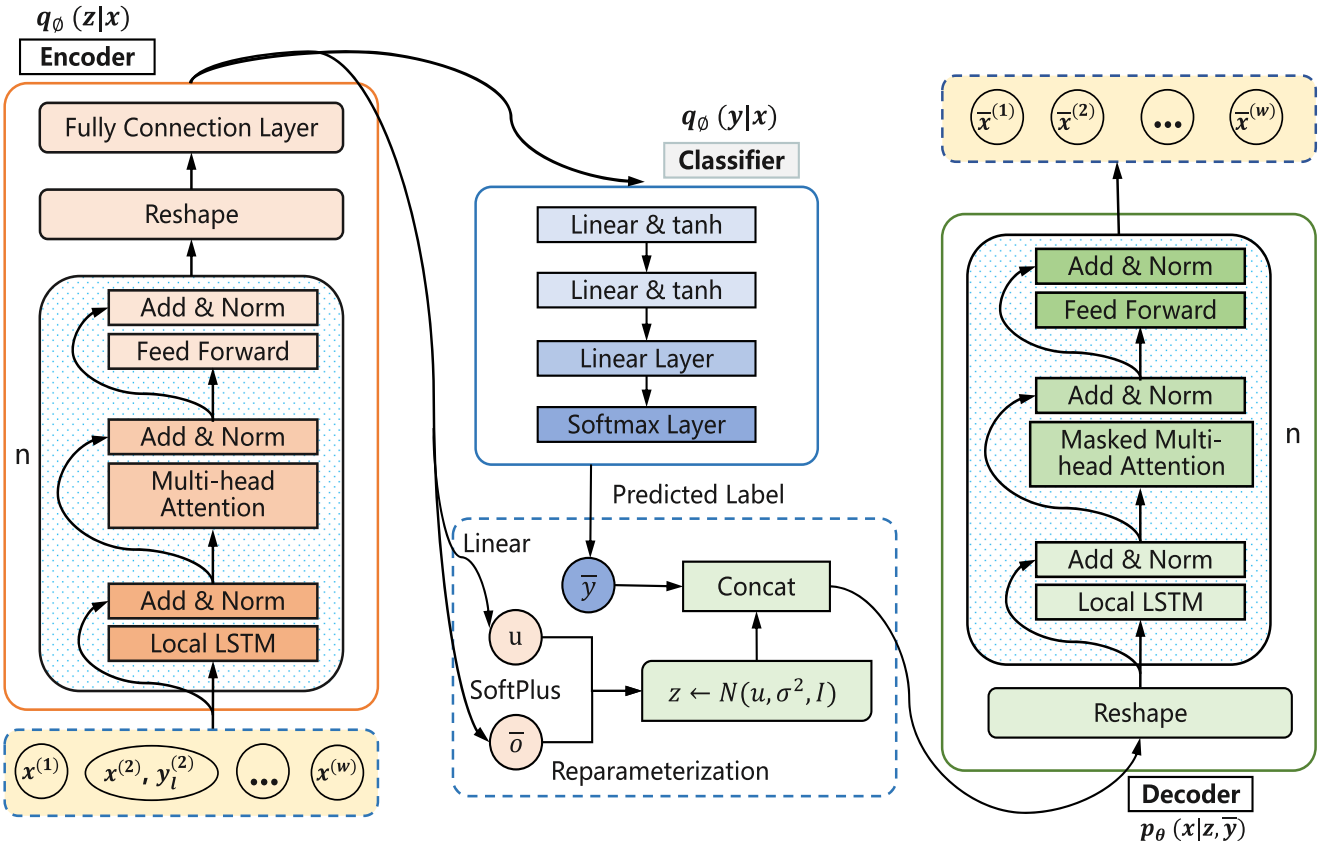


Fig. 5 Network Structure of RT-SemiVAE

weak local position signal is not needed. RT-SemiVAE uses the LSTM to model short-term dependencies of multivariate time series and the transformer to learn long-term dependencies. The entire network structure is shown in Fig. 5, and in the decoder ($p_\theta(x|z, \overline{y})$), the encoder-decoder attention layer in the standard transformer decoder is removed. Because the deep generation network VAE is used to sample from random variable z, it is no longer necessary to use the features learned by the encoder ($q_\phi(z|x)$) to calculate the attention of the encoder-decoder attention layer.

3.3.2 Online detection process

RT-SemiVAE trains the anomaly detection model offline according to the set time interval to capture the change characteristics of entity multivariate KPIs in time. While training RT-SemiVAE, to achieve good reconstruction, the input data (x, y_l) is normalized to a normal distribution by Z-Normalization. Then, We improve the $ELBO$ loss function to make the anomaly detection model pay attention to normal pattern and ignore the abnormal pattern as much as possible during training, which makes the reconstruction probability of the corresponding data points low; thus, the model can detect anomalies more easily. In addition, RT-SemiVAE uses a small amount of labeled data to drive a large amount of unlabeled data to train the $Classifier$ and uses the labels predicted by the $Classifier$ for the decoding reconstruction process to improve anomaly detection accuracy.

During RT-SemiVAE anomaly detection, if the KPIs monitoring value of an entity at a certain time (i.e., $x^{(t)}$ at time t) is abnormal, it indicates that the entity is abnormal, and vice versa. RT-SemiVAE still uses the reconstruction probability independent of the data distribution as the anomaly detector, and uses the Monte Carlo method [14] to approximate the solution, as detailed in (5). Then, the higher the reconstruction probability score $r^{(t)}$ is, the better the reconstruction effect is, and the more likely $x^{(t)}$ is to be judged as normal. Finally, the health state of the entity is judged according to the set threshold. If the reconstruction probability score $r^{(t)}$ at time t is less than the set threshold, then $x^{(t)}$ is abnormal at time t, and the corresponding service or application is also abnormal. Otherwise, the corresponding service or application is judged to be in a normal state. The entire anomaly detection process of RT-SemiVAE is expressed by Algorithm 3.

Algorithm 3 RT-SemiVAE anomaly detection algorithm.

Input: training data $(x_{train}, y_l^{train}) = ((x^{(1)}, y_l^{(1)}), (x^{(2)}, y_l^{(2)}), \cdots, (x^{(train)}, y_l^{(train)}))$, the $Threshold$, online testing data $x_{test}=(x^{(1)}, x^{(2)}, \cdots, x^{(test)})$, local window length W.

Output: abnormal results ar.

$Classifier$, $Encoder$, $Decoder$ ← initialize network parameters;
(x_{train}, y_l^{train}) ← Z-Normalization(x_{train}, y_l^{train});
Encoder, Decoder, Classifier ← Train the RT-SemiVAE network model using the multivariate time series;
x_{test} ← Z-Normalization (x_{test});
while x_{test} *data have a value* **do**
  $h_f^e \leftarrow Encoder(x_{test}, W)$;
  $\mu \leftarrow W_u^T f_\phi(h_f^e) + b_u, \sigma \leftarrow SoftPlus[W_\sigma^T f_\phi(h_f^e) + b_\sigma] + \varepsilon$;
  for $l \leftarrow 1$ *to* L **do**
    $z^l \leftarrow N(\mu, \sigma^2, I)$;
    $(\mu_{\overline{x}^l}, \sigma_{\overline{x}^l}) \leftarrow Decoder(concat(z^l, Classifier(x_{test})))$;
    $reconstructionProbability^l \leftarrow p_\theta(x_{test}|\mu_{\overline{x}^l}, \sigma_{\overline{x}^l})$;
  $rp^{(t)} \leftarrow \frac{1}{L}\sum_{l=1}^{L} reconstructionProbability^l$;
  $r^{(t)} \leftarrow Sigmoid(rp^{(t)})$;
  `// Reconstruction probability score at time` t
  if $r^{(t)} < threshold$ **then**
    $ar^{(t)} \leftarrow 1$;
    `//` $ar^{(t)}$ `represents the abnormal result at time` t`, 0 represents normal, and 1 represents abnormal.`
  else
    $ar^{(t)} \leftarrow 0$;
return ar;

Due to the complex dependencies between services or applications, the anomalies of a service often cause anomalies in the associated services, and then, a series of abnormal alarms are generated. Therefore, it is unreasonable to judge the earliest abnormal services or applications as real abnormal entities, and the discovery of root cause entities will also be delayed. Therefore, it is necessary to trace and locate the root causes of anomalies.

3.3.3 Anomaly location process

The construction time and update frequency of the service dependency graph in different cloud environments are different. Considering that the construction process may produce long delays, which affect the real-time accuracy of anomaly location, we do not consider a few application scenarios with high real-time requirements for fault location. Conversely, to ensure the real-time accuracy of anomaly location as much as possible, it is necessary to control the delay overhead of service dependency graph construction in an acceptable range.

When tracing the true anomalous entities, RT-SemiVAE leads to the concept of a service call chain based on the constructed service dependency graph, which traces the anomalous entity through the call relationship between services. The service call chain defined by RT-SemiVAE consists of all the call relationships with a service as the endpoint; that is, the dotted box in Fig. 6 represents the service call chains corresponding to this service.

The acquisition of the service call chains can be divided into three steps:

(1) Obtain all the service call chains. By querying the graph database, RT-SemiVAE can obtain the call relationships of all services in the cloud environment, and identify the irrelevant subservices with a unique ID. This operation helps to ensure quick retrieval when anomalies occur.
(2) Delete unnecessary call relations. Some service call chains are not necessary to retain because they are included in longer service call chains. For example, the path of $ServiceA \rightarrow ServiceC$ in Fig. 6 is contained in the $ServiceA \rightarrow ServiceC \rightarrow ServiceD$ and $ServiceA \rightarrow ServiceC \rightarrow ServiceE$ call chains.

(3) Group all service call chains according to the starting service endpoint. The service call chain defined by our RT-SemiVAE is all the service call relations on the right side of Fig. 6.

RT-SemiVAE achieves indicator-level anomaly location by selecting several single-indicator time series with the highest reconstruction probability as the root causes of anomalies. Therefore, we must obtain the reconstruction probability of all dimensions $(x_1^{(t)}, x_2^{(t)}, \cdots, x_M^{(t)})$ in $x^{(t)}$. RT-SemiVAE calculates the reconstruction probability of M-dimensional $x^{(t)}$, and $p_\theta(x^{(t)}|z_{(t-w+1:t)})$ is constrained to $N(\mu_{x^{(t)}}, \sigma^2_{x^{(t)}}, I)$, where μ is the mean, σ^2 is the variance and I is the parameter of the multivariate normal distribution $N(0, I)$. Therefore, $p_\theta(x^{(t)}|z_{(t-w+1:t)}) = \prod_{i=1}^{M} p_\theta(x_i^{(t)}|z_{(t-w+1:t)})$ holds, and (6) holds for further derivation:

$$\log(p_\theta(x^{(t)}|z_{(t-w+1:t)})) = \sum_{i=1}^{M} \log(p_\theta(x_i^{(t)}|z_{(t-w+1:t)})) \quad (6)$$

The reconstruction probability score $r^{(t)}$ at time t can be expressed by $\sum_{i=1}^{M} r_i^{(t)}$, that is, $r^{(t)} = \sum_{i=1}^{M} r_i^{(t)}$ holds, where $r_i^{(t)}$ is the ith-dimensional reconstruction score of $x^{(t)}$, $r_i^{(t)} = Sigmoid(\frac{1}{L}\sum_{l=1}^{L} \log(p_\theta(x_i^{(t)}|z_{t-w+1:t}^l)))$. For an abnormal $x^{(t)}$, RT-SemiVAE explains it through the abnormal scores of each dimension of $x^{(t)}$, the smaller $r_i^{(t)}$ is, the more likely it is to cause $x^{(t)}$ to be abnormal. RT-SemiVAE focuses on several dimensions with small abnormal scores to achieve rapid localization and repair of anomalies. The entire process of RT-SemiVAE anomaly localization is summarized in Algorithm 4.

Algorithm 4 RT-SemiVAE anomaly localization algorithm.

Input: Abnormal entity set $Entities[n]$, service dependency relation set $\{SD\}$.
Output: Root entity $rEntity$, ordered indicator set of root entities $SortedIndex$.

```
/* Obtains the service call chains for each exception entity.                    */
chain ← Array[n], lastEntity ← Array[n]; //initialization;
for i ← 0 to n − 1 do
    while Entities[i] ∈ {SD} and has the associated service invocation chains do
        chain[i]. Add(service invocation chain);
    chain[i]. Delete(repeated invocation relationships);
/* Obtains the last service entity in the service call chains.                   */
for i ← 0 to n − 1 do
    for j ← 0 to length(chain[i]) do
        if chain[i][j + 1] == NULL then
            lastEntity[i].Add(chain[i][j]);
        else
            j++;
rEntity ← getRootEntity(Entities, lastEntity);
// Obtaining the root entity rEntity
/* Calculate the reconstruction probability scores of all dimensions of the
   root entity, where M is the dimension                                         */
for i ← 1 to M do
    r_i^(t) = Sigmoid(1/L Σ_{l=1}^{L} log(p_θ(x_i^(t)|z_(t−w+1:t)^l)));
/* Sorting all index dimensions from small to large according to their
   reconstruction probabilities. The previous index shows a greater
   contribution to the x^(t) anomaly, and the latter shows a smaller impact on
   the x^t anomaly.                                                              */
sortedIndex ← positiveSort(r_i^(t)), i ∈ 1, 2, ···, M;
return rEntity, sortedIndex;
```

In Algorithm 4, RT-SemiVAE first obtains the service call chains for each abnormal entity based on the abnormal entity set $Entities[n]$ and service dependencies $\{SD\}$. Because the proposed service call chain is a call relationship

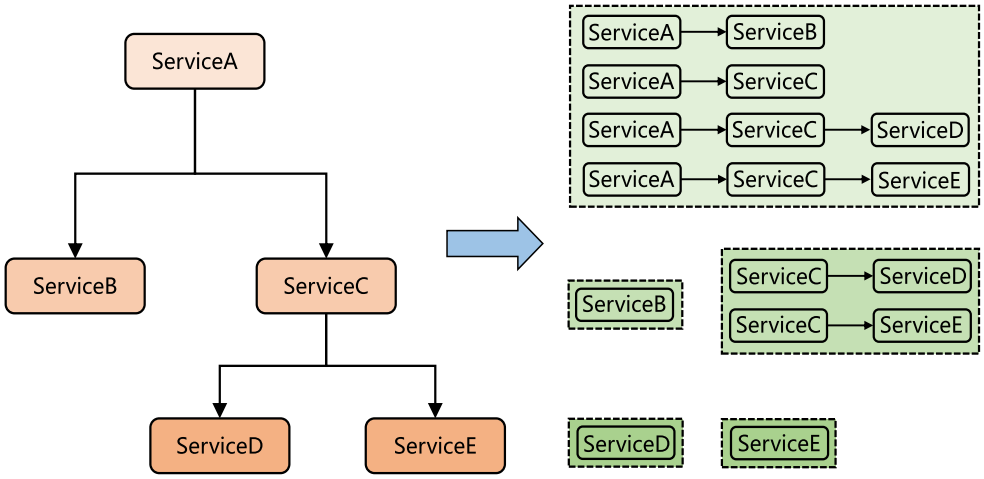


Fig. 6 Example of a service dependency graph converting to service call chains

between all services or applications with a service as the endpoint, all the end entities on service call chains for each abnormal entity are represented as $lastEntity$. Then, the root entity $rEntity$ is determined according to the $lastEntity$ and the $Entities[n]$, and the reconstruction probability scores of all indicators of the root entity are calculated. Finally, the indicators of the root entity are arranged from small to large according to the corresponding reconstruction probability scores, and the sorted indicator list $sortedIndex$ is obtained. Then locate the root causes of the anomaly at the indicator level.

In summary, RT-SemiVAE first obtains the complex dependencies between services or applications through a service registration mechanism and then judges the abnormal state of entities according to the multivariate time series of entities. This method uses the VAE with random variables to learn the complex distribution of data, leverages the Local-LSTM to learn the local dependence of data, and uses the transformer to learn the long-term dependence. Finally, RT-SemiVAE traces the root entities that cause anomalies through the service call chains and performs fine-grained localization of anomaly causes at the indicator level. RT-SemiVAE has absolute advantages in capturing the time dependence and complex distribution of data, and only a small amount of labeled data are necessary to achieve good anomaly detection performance. This method can thus meet the timeliness requirements of large-scale time series anomaly detection and location in a cloud environment.

4 Experiments and evaluation

In this section, we validate the LR-SemiVAE and RT-SemiVAE at the algorithm level (Section 4.1) and application level (Section 4.2). In Section 4.1, we introduce the design of the algorithm experiment, data sets and anomaly detection performance evaluation and verify the effectiveness of introducing a parallel multihead attention mechanism. Then, we design the MAD-VAE prototype system to verify the anomaly detection and location performance of LR-SemiVAE and RT-SemiVAE in a real cluster environment. The prototype architecture and application performance evaluation are introduced in Section 4.2. Finally, we summarize the experiment in Section 4.3.

4.1 Algorithm and strategy verification

4.1.1 Experimental design and data sets

Because we use a semisupervised learning method, it is necessary to consider setting the proportion of labeled data in the experiment. For different anomaly detection scenarios, different $classifiers$ and data sets with different probability distributions, the set proportions are not the same. If the proportion of labeled data is high, the accuracy of anomaly detection will be improved, but the model training time and difficulty in obtaining labeled data will also increase, and vice versa. Therefore, we verified the anomaly detection performance of the LR-SemiVAE and RT-SemiVAE strategies by controlling variables in the experiments; thus, the problem of setting the proportions does not affect the innovative contribution of this work.

We used the Yahoo public data set[1] to evaluate the anomaly detection performance of LR-SemiVAE. The data set is composed of real traffic data and synthetic data, including 100 indicator data collected by different sensors, with a total of 1680 samples. The data set consist of a multivariate time series (100 dimensions) with noise, seasonality and different trends. First, the data set is

[1] https://webscope.sandbox.yahoo.com/catalog.php?datatype=s&did=70

arranged in a matrix according to the temporal and feature dimensions. Each column represents the KPI data of a sensor at different time points, and each row represents the KPI data of all sensors at the current time point. The last column is the data label corresponding to the current time point t. Second, because the Yahoo data set only provides the label of a single indicator and does not consider the abnormal situation of the entire entity, if all KPIs of the entity are normal at a certain time, it is judged as normal for the entity; otherwise, it is judged as abnormal. Finally, because the time interval of data acquisition is 60 minutes, the length of sliding window w (length of the historical subsequence fragment) is set to $w = 3 * i, i = 1, 2, 3, \cdots, 10$, and the dimension of the potential random variable z should be set to 8 on this data set [12]. LR-SemiVAE is implemented on a GPU using PyTorch 1.3.1. Based on empirical learning, the initial learning rate of the adaptive moment estimation (ADAM) optimizer is usually set to 0.001 and $\lambda = 0.1L_{label}$, where L_{label} is the number of labeled samples in the current data set.

To evaluate the anomaly detection performance of LR-SemiVAE and RT-SemiVAE, we selected and implemented five baseline algorithms for comparative experiments. They are MAD-GAN [9], LSTM-VAE [10, 11], VAE-GAN [12], VAE [28] and semisupervised VAE (VAE M2) [29]. The following three indicators are often used to evaluate anomaly detection performance:

(1) **Accuracy rate:** represents the accuracy of the model prediction results:

$$precision = \frac{TP}{TP + FP} \tag{7}$$

(2) **Recall rate:** represents the ratio of the positive cases found in the model to the real positive cases:

$$recall = \frac{TP}{TP + FN} \tag{8}$$

(3) **Comprehensive indicators:** accuracy rate and recall rate weighted harmonic average:

$$F1-score = \frac{2 \cdot precision \cdot recall}{precision + recall} \tag{9}$$

TP (True Positive: when the real label $L_t = 1$, the detected label $D_t = 1$) represents the correct detection of anomalies, FP (False Positive: when $L_t = 0$, $D_t = 1$) is the inaccurate detection of anomalies, TN (True Negatives: when $L_t = 0$, $D_t = 0$) represents the correct allocation of normal, FN (False Negative: when $L_t = 1$, $D_t = 0$) is the inaccurate allocation of normal, and the higher the $F1-score$ is, the better the anomaly detection performance is.

4.1.2 Evaluation of anomaly detection performance of LR-SemiVAE

Most time series anomaly detection algorithms based on VAEs select all abnormal samples and the same number of normal samples as the test data set and the remaining samples as the training data set. Because selecting normal data to train the model can make the reconstruction probability score of the model low when encountering abnormal data (if the reconstruction probability is less

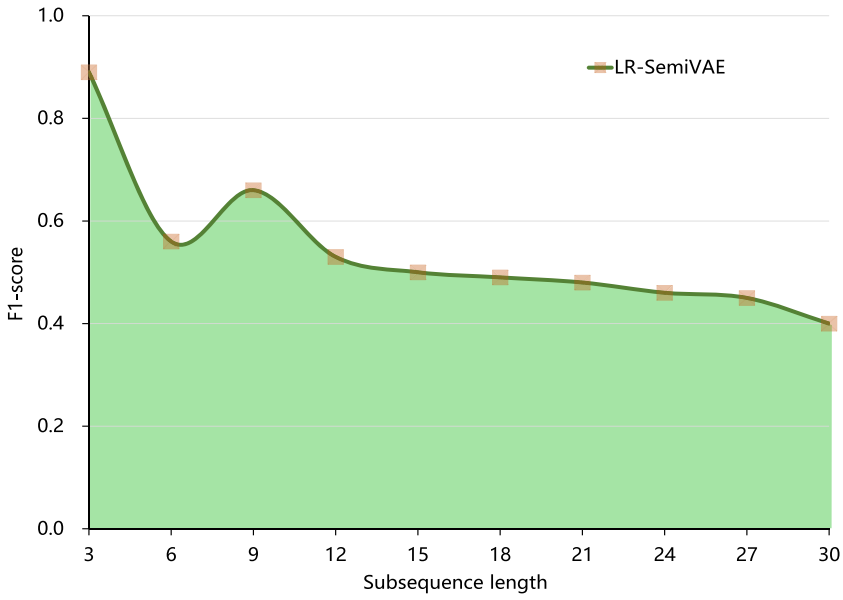
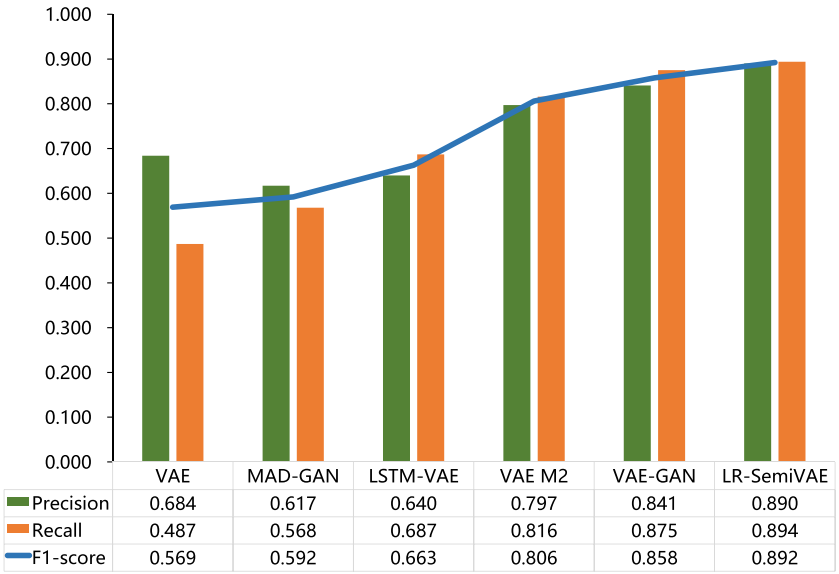


Fig. 7 $F1-score$ values under different window lengths (w)



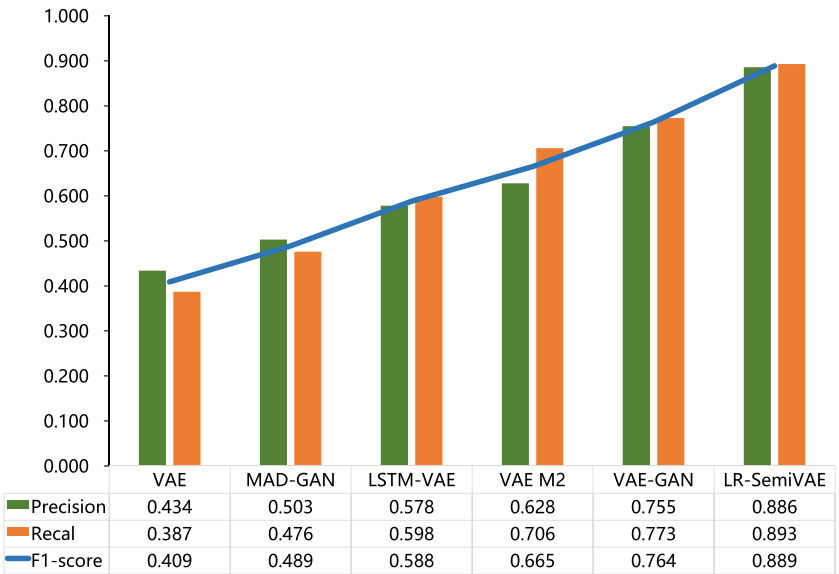
	VAE	MAD-GAN	LSTM-VAE	VAE M2	VAE-GAN	LR-SemiVAE
Precision	0.684	0.617	0.640	0.797	0.841	0.890
Recall	0.487	0.568	0.687	0.816	0.875	0.894
F1-score	0.569	0.592	0.663	0.806	0.858	0.892

Fig. 8 Anomaly detection performance evaluation of LR-SemiVAE with all normal data training

than the threshold, it is judged as abnormal), the accuracy of anomaly detection improves. To determine the sliding window length w, we divided the training data set into multiple sets of different window lengths ($w = 3 * i, i = 1, 2, 3, \cdots, 10$). The subsequence fragments were used for model training, and the corresponding $F1 - score$ value on the test data set was recorded (Fig. 7) to obtain the optimal window length of the current data set (maximum $F1 - score$). The RT-SemiVAE strategy and other baseline algorithms are the same.

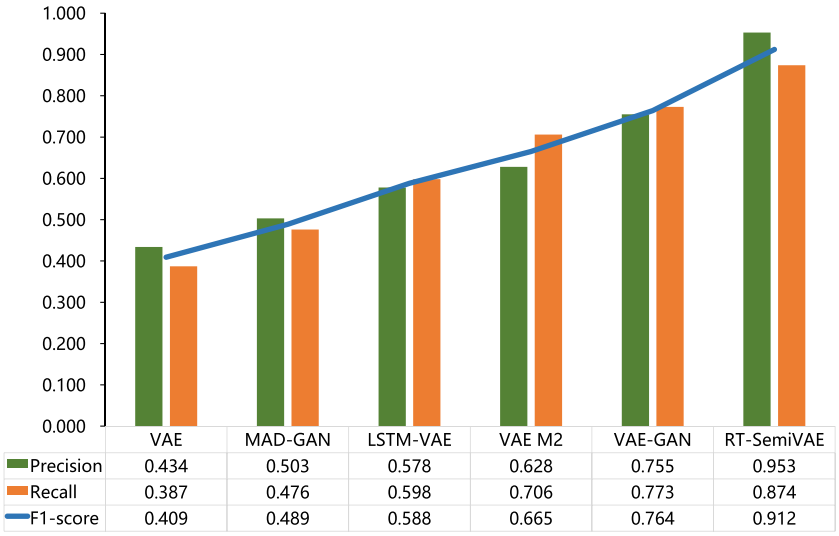
Figure 7 shows that LR-SemiVAE performed the best when the window length is $w = 3$. Similarly, the baseline algorithms were also trained with all normal data to obtain the corresponding optimal window length, which is 3 (the same as that of LR-SemiVAE). Then, LR-SemiVAE was compared with the baseline algorithms for anomaly detection performance in the case of all normal data training. The results are shown in Fig. 8.

Figure 8 shows that the $F1 - score$ value of LR-SemiVAE is 3.96% higher than that of VAE-GAN, 10.67% higher than that of VAE M2 and 34.54% higher than that of LSTM-VAE. To evaluate the anomaly detection performance of LR-SemiVAE under training with data that are not all normal, the first 75% of the Yahoo data set was used for training, and the last 25% was used for testing. The amount of extracted labeled data is 40% of the test data set. Based on these experimental conditions, the baseline models were retrained, and then, the test data set was used for anomaly detection. The final detection performance comparison results are shown in Fig. 9.



	VAE	MAD-GAN	LSTM-VAE	VAE M2	VAE-GAN	LR-SemiVAE
Precision	0.434	0.503	0.578	0.628	0.755	0.886
Recal	0.387	0.476	0.598	0.706	0.773	0.893
F1-score	0.409	0.489	0.588	0.665	0.764	0.889

Fig. 9 Anomaly detection performance evaluation of LR-SemiVAE with abnormal data training



	VAE	MAD-GAN	LSTM-VAE	VAE M2	VAE-GAN	RT-SemiVAE
Precision	0.434	0.503	0.578	0.628	0.755	0.953
Recall	0.387	0.476	0.598	0.706	0.773	0.874
F1-score	0.409	0.489	0.588	0.665	0.764	0.912

Fig. 10 Anomaly detection performance evaluation of RT-SemiVAE with abnormal data training

Comparing Figs. 8 and 9 shows that adding abnormal data in the training process of the baseline algorithms interferes with the reconstruction effect and causes the abnormal detection performance to seriously decline. However, the $F1-score$ value of the LR-SemiVAE model trained with abnormal data only changes by 0.34% in the anomaly detection, and the anomaly detection performance remains basically unchanged, which demonstrates the effectiveness of LR-SemiVAE in improving the $ELBO$ (maximum variational lower bound) loss function. Figure 9 also shows that in the case of including abnormal data in training, the $F1-score$ comprehensive indicator of LR-SemiVAE is 16.36% higher than that of VAE-GAN, 33.68% higher than that of VAE M2 and 51.19% higher than that of LSTM-VAE anomaly detection algorithms.

4.1.3 Evaluation of anomaly detection performance of RT-SemiVAE

To verify the anomaly detection performance of RT-SemiVAE, the Yahoo data set was also used to compare with the VAE, MAD-GAN, LSTM-VAE, VAE M2, and VAE-GAN baseline algorithms in terms of the accuracy rate, recall rate, and $F1-score$. When using all normal data for training, the window length implemented in RT-SemiVAE to obtain the best $F1-score$ is 21, while the window length implemented in VAE, MAD-GAN, LSTM-VAE, VAE M2, and VAE-GAN algorithms is 3. Then, the first 75% of the Yahoo data set was used for training, and the last 25% of the data set was used for testing to evaluated the anomaly detection performance of the baseline algorithms and

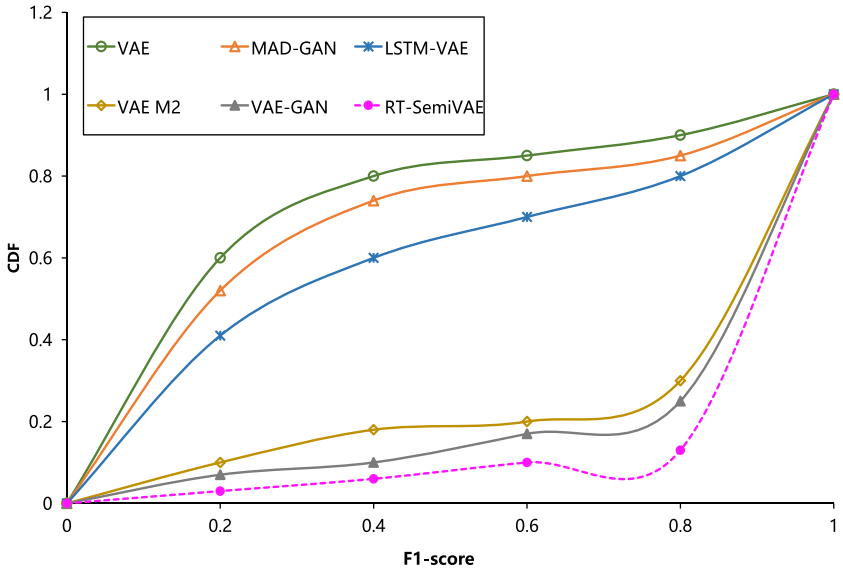


Fig. 11 Evaluation of CDF indicators for optimal $F1-score$ values

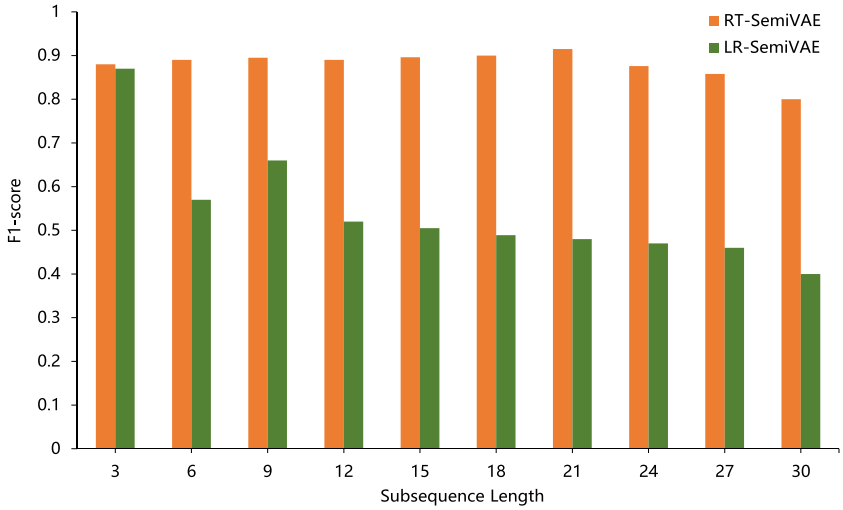


Fig. 12 Effect of the attention mechanism on the $F1-score$ value in RT-SemiVAE

RT-SemiVAE with the best window length and abnormal data involved in training. The final accuracy rate, recall rate, and $F1-score$ values are shown in Fig. 10.

Figure 10 shows that the $F1-score$ comprehensive indicator of RT-SemiVAE is 19.37% higher than that of VAE-GAN, 37.14% higher than that of VAE M2 and 55.10% higher than that of LSTM-VAE. In addition, we also verified the cumulative distribution function (CDF) of the best $F1-score$ of VAE, MAD-GAN, LSTM-VAE, VAE M2, VAE-GAN and RT-SemiVAE. The results are shown in Fig. 11. In the CDF figure, the closer the curve is to the upper left corner, the worse the anomaly detection performance is. The closer the curve is to the lower right corner, the better the anomaly detection performance is.

Figure 11 shows that the RT-SemiVAE, VAE-GAN, VAE M2 algorithms based on semisupervised learning achieve good anomaly detection performance, with 70% of the best $F1-score$ values exceeding 0.8. Their anomaly detection performances are much better than those of LSTM-VAE, MAD-GAN and VAE. There are two primary reasons for this result: 1) LSTM-VAE, MAD-GAN and VAE are unsupervised learning methods, and no labels are introduced in the training process; and 2) VAE uses a feedforward neural network to encode and decode, without considering the time dependence of the sequence.

4.1.4 Verification of the multihead parallel attention mechanism.

To verify the effect of parallel multihead attention mechanism transformer introduced by the RT-SemiVAE strategy on anomaly detection performance, we compared the performance of LR-SemiVAE and RT-SemiVAE with abnormal data training on the Yahoo data set. Figure 12 shows the $F1-score$ value comparison of the two strategies under different subsequence lengths ($w = 3 * i, i = 1, 2, 3, \cdots, 10$).

Figure 12 shows that RT-SemiVAE with the attention mechanism achieves the best performance at $w = 21$, which is seven times the length of the subsequence when LR-SemiVAE obtains the best performance. The anomaly detection models trained with subsequences of different lengths in RT-SemiVAE have better performance than those in LR-SemiVAE, and the anomaly detection performance of RT-SemiVAE is relatively stable. However, after the subsequence reaches a certain length, the anomaly detection performance of LR-SemiVAE gradually decreases with the increase of the subsequence length. Because the transformer introduced in RT-SemiVAE is a parallel multihead attention mechanism, we further verified its impact on the model training speed, and the specific results are shown in Fig. 13.

Figure 13 shows that the RT-SemiVAE strategy based on the transformer converges faster than the LR-SemiVAE strategy, which only uses the LSTM for encoding and decoding. When the number of iterations is approximately

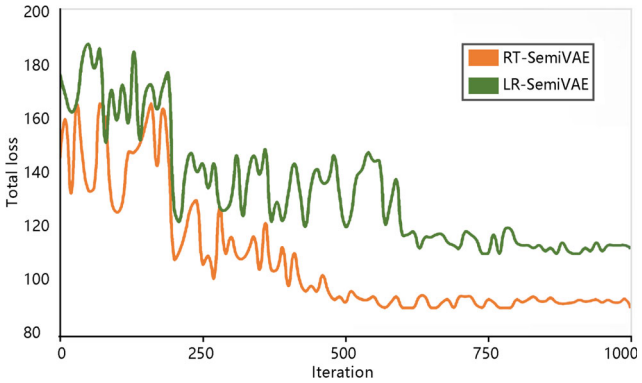


Fig. 13 Comparison of total loss between RT-SemiVAE and LR-SemiVAE at different iteration times

500, the loss value of RT-SemiVAE training is basically stable, while the LR-SemiVAE strategy begins to converge after approximately 800 iterations. The introduction of a parallel multihead attention mechanism in RT-SemiVAE thus accelerates the model training speed, reducing training time.

4.2 Prototype verification

In Section 4.2, the LR-SemiVAE strategy for a single application and the RT-SemiVAE strategy for long-term service dependency were integrated to construct the prototype verification framework. Their anomaly detection performance and positioning accuracy were evaluated by building container clusters, deploying service dependence applications, simulating access requests, and injecting anomalies to verify their performance and applicability in a real cluster environment.

4.2.1 Verification framework and experimental environment

We designed and developed a prototype system MAD-VAE: the system framework is shown in Fig. 14. First, we built a container cluster using the Kubernetes technique that consists of a master management node and multiple working nodes. Second, we use the open source Prometheus monitoring system [30] to replace the original cAdvisor monitoring system of Kubernetes. The Prometheus has a service discovery mechanism, which can automatically monitor the real-time state of the cluster through the API server; thus, it can be applied to the KPI monitoring of the Kubernetes cluster system. Given that the data in Prometheus cannot exist locally for too long, the KPI data collected by the monitoring system were organized by the data aggregation module according to entities (e.g., servers, containers, services) and time; then, the multivariate KPIs

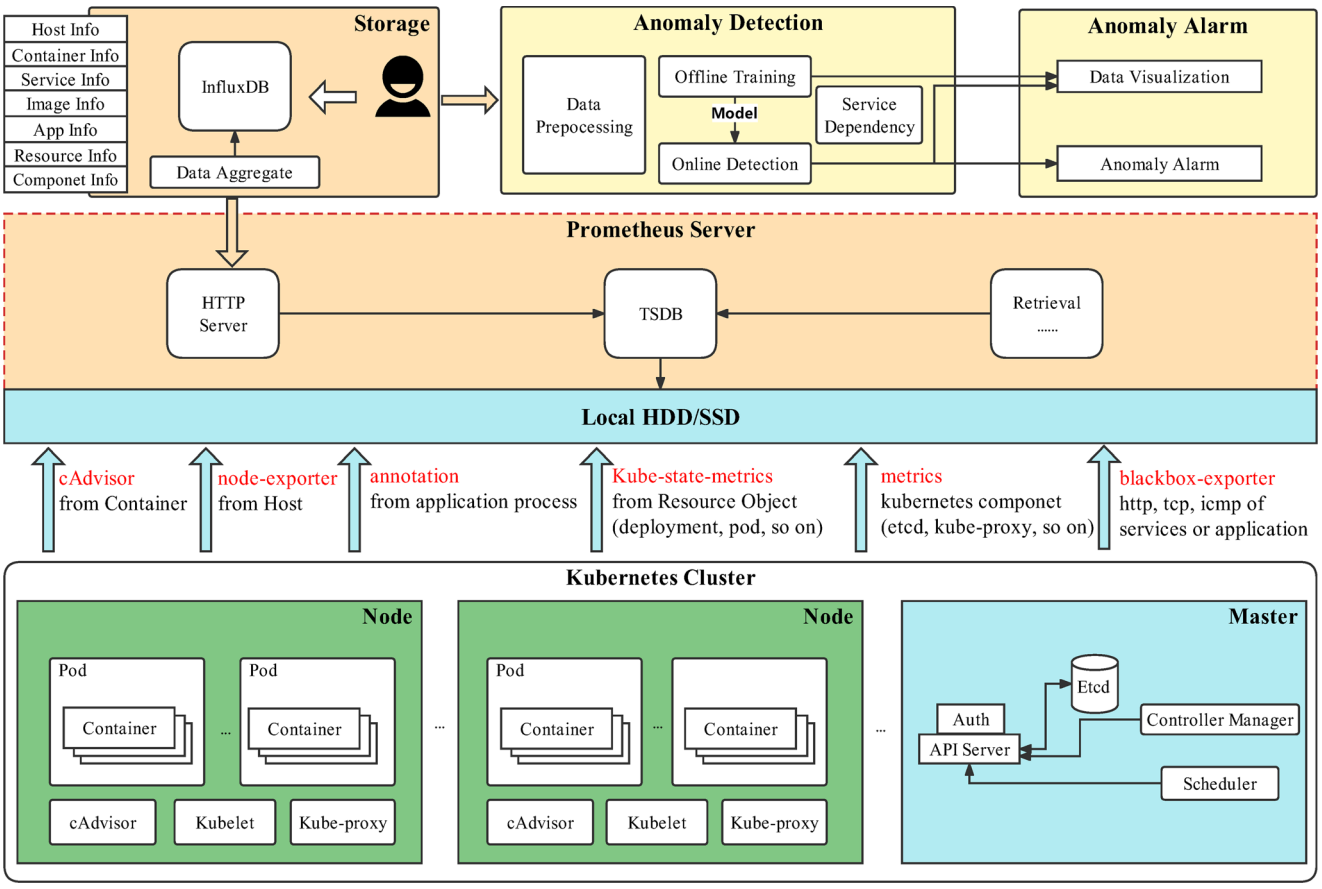


Fig. 14 Architecture of the MAD-VAE prototype system

Table 1 Comparisons of MAD-VAE and Prometheus

Comparative content	Prometheus system	MAD-VAE prototype
Expandability	Supporting large-scale clusters.	Supporting larger-scale clusters and faster detection speed.
Detecting object	Single indicator	Entities (hosts, containers, services, etc.).
Anomaly location	No. It must integrate link tracking components, such as Zipkin, Pinpoint, central application tracking (CAT), and cannot perform fine-grained location of anomalies after integration.	Yes. It can trace the root entities and locate the causes of anomalies at the indicator level.

were stored in the time series database InfluxDB [31]. In this process, we labeled some data according to domain knowledge for anomaly detection and location. Then, all the KPI data of entities in InfluxDB were extracted and preprocessed for offline training, and the trained model was used for online real-time detection to determine whether an entity is abnormal. After an anomaly occurs, the root causes of the anomaly were tracked according to the service dependence relations.

The MAD-VAE prototype system replaced the anomaly detection and alarm module in Prometheus with LR-SemiVAE and RT-SemiVAE strategies. Compared with the original anomaly detection mechanism, the innovation of the MAD-VAE prototype system is that it can perform entity-oriented anomaly detection; thus, anomaly detection is more accurate and efficient, and MAD-VAE can also accurately locate the root entities and their abnormal indicators that cause anomalies. More comparisons are shown in Table 1.

Because the prototype verification data were collected in minutes, the dimension of the potential variable z (random variable) was set to 64, the dimension of the label predicted by the $Classifier$ was set to 2, and the ADAM optimizer with an initial learning rate of 0.001 was used for training. The length of the slide window was set to $w = 30 * i, i = 1, 2, \cdots, 10$, and each attention mechanism consists of three identical components ($N = 3$). The RT-SemiVAE encoder and decoder in MAD-VAE have four parallel attention mechanisms. Then, 15 servers were used to build the Kubernetes cluster and deploy the Prometheus monitoring components, the distributed time series database InfluxDB, the Eureka registration Center, the Neo4j graph database and eight microservice instances of Sock Shop v2.0[2]. Each service used a node to deploy, and the specific configuration is shown in Table 2.

In Table 2, each node limited the speed of the network interface card by installing Wondershaper, where 2 Mbps and 8 Mbps correspond to the maximum download speeds of 256 KB/s and 1 MB/s, respectively. The upload speed was set to 1/4 of the download speed according to the standard. We used the QEMU Virtual CPU (CPU64-rhel7)) whose main parameter is the number of processor cores, including the number of processors (P) and the cores (C) of each processor. For example: "2P 4C" means there are eight cores.

We wrote a simulation program to access the Sock Shop application through the entry service Front-end, and simulated the concurrent requests and access frequency at different times. In this process, the ChaosBlade[3] fault generator was used to simulate three common types of anomalies: packet loss, memory leakage and CPU occupation. By improving ChaosBlade, different lengths of activation modes are used for each anomaly type: 1) linear mode, where the frequency of the same type of anomalies increases linearly with time; 2) exponential mode, where the frequency of anomalies increases exponentially with time; and 3) random mode, where the frequency of abnormal occurrences is irregular. The anomaly detection and localization performance of MAD-VAE was evaluated by comparing with Prometheus from three aspects: anomaly types, activation modes, and anomaly services.

4.2.2 Performance evaluation of anomaly detection

To evaluate the anomaly detection performance of MAD-VAE in the simulated cloud environment, we used packet loss, memory leakage and CPU hog to simulate the Sock Shop application, and used Prometheus system to monitor, collect and process data, so as to record the real abnormal data, alarm data, training samples, test samples and the total request. Then the collected multivariate time series samples were used for MAD-VAE offline training. Finally, the trained model was used to detect the labeled test samples. Figure 15 shows the changes in the $F1 - score$ comprehensive indicators of MAD-VAE and Prometheus within 120 minutes after the online detection phase begins.

Figure 15 shows that the $F1 - score$ of MAD-VAE and Prometheus in detecting packet loss anomalies is lower than that of memory leakage and CPU hog. This result is likely due to the excessive network speed during the propagation of packet loss anomalies, which makes it difficult for MAD-VAE and Prometheus to capture the anomalies,

[2] https://github.com/microservices-demo/microservices-demo

[3] https://github.com/chaosblade-io/chaosblade

Table 2 Configuration of Kubernetes cluster

Host	Memory	CPU	Bandwidth	Disk	Main Deployment
Master	2G	2P 4C	8 Mbps	100G	
Node1	4G	2P 4C	8 Mbps	500G	Prometheus Service
Node2	4G	2P 4C	8 Mbps	500G	InfluxDB
Node3	4G	2P 4C	8 Mbps	500G	InfluxDB
Node4	4G	2P 4C	8 Mbps	500G	InfluxDB
Node5	2G	2P 4C	8 Mbps	100G	Cart Service
Node6	4G	1P 2C	4 Mbps	500G	Catalogue Service
Node7	4G	2P 4C	8 Mbps	100G	Order Service
Node8	2G	2P 4C	8 Mbps	100G	Front-end Service
Node9	2G	1P 2C	4 Mbps	500G	User Service
Node10	4G	2P 4C	8 Mbps	100G	Payment Service
Node11	2G	1P 2C	4 Mbps	100G	Shipping Service
Node12	4G	2P 4C	8 Mbps	500G	Queue-Master Service
Node13	4G	2P 4C	8 Mbps	100G	Eureka
Node14	4G	1P 2C	8 Mbps	500G	Neo4j

so that its detection performance decreases. However, for detecting memory leak anomalies, its performance is unstable. Further research find that its recall rate shows a downward trend, so the $F1-score$ is unstable when detecting memory leak anomalies. However, the anomaly detection performance of MAD-VAE and Prometheus tends to be stable on CPU hog anomalies. In conclusion, MAD-VAE performs better than Prometheus in detecting all three anomaly types in random activation mode. To further verify the anomaly detection performance of MAD-VAE, linear, exponential and random activation modes were used to simulate the frequency of CPU hog anomalies. Figure 16 shows the comparison of $F1-score$ within 120 minutes after the start of online detection under three different activation modes.

Figure 16 shows that Prometheus and MAD-VAE achieve good stability in the random activation mode. In the linear activation mode, the $F1-score$ of Prometheus exhibits a downward trend with time, and MAD-VAE decreases marginally but tends to be relatively stable. In the exponential activation mode, the $F1-score$ of the two approaches fluctuates greatly. Overall, MAD-VAE still has performance advantage over Prometheus under three different activation modes.

The previous part evaluates the anomaly detection performance of the entire Sock Shop application under different anomaly types and activation modes without considering the internal services of Sock Shop. Therefore, the following part evaluates the performance of Order, Catalogue, Payment, and Cart services, which are the core of the Sock Shop application, and uses them as the target services for anomaly injection to evaluate the anomaly detection performance of a single service. The $F1-score$ in Fig. 17 is calculated with service dependencies. The frequent communication of the Order service increases the difficulty of anomaly detection and

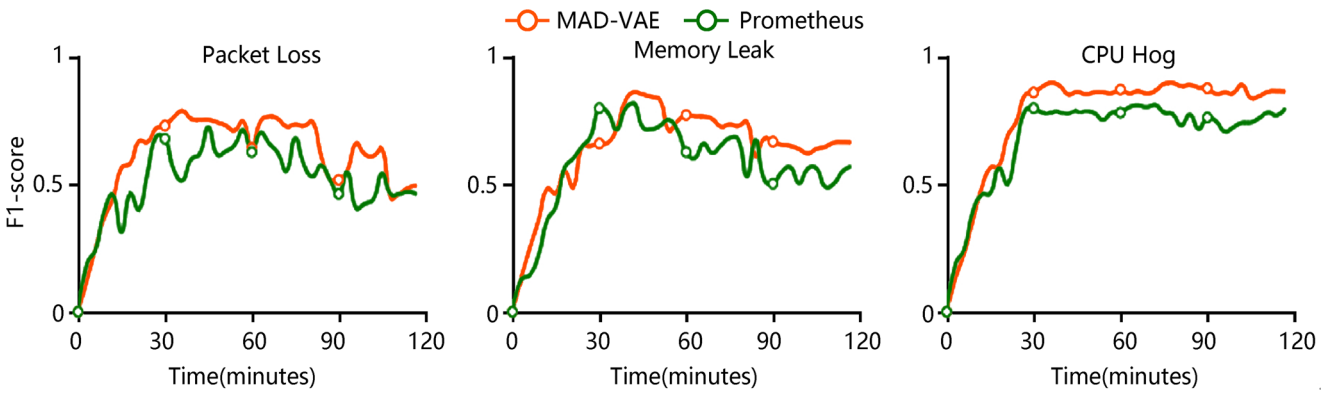


Fig. 15 Comparison of $F1-score$ values under different anomaly types

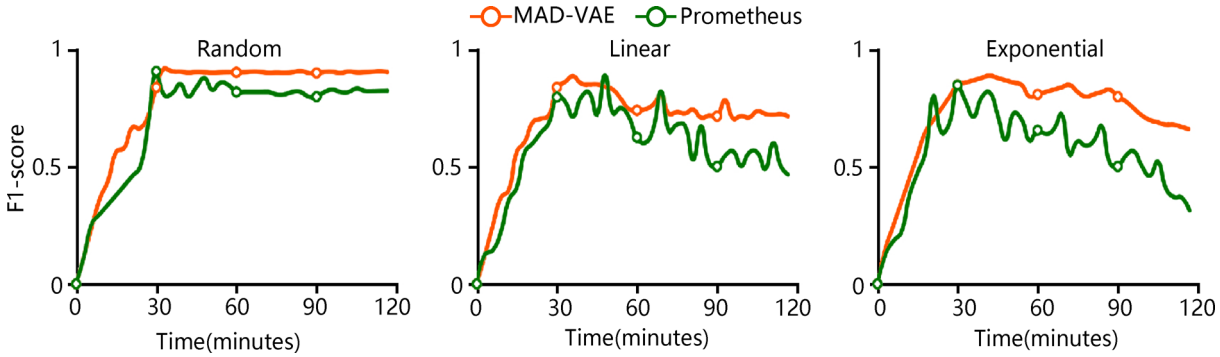


Fig. 16 Comparison of $F1-score$ values under different activation modes

makes the performance unstable. However, for the four services of Order, Catalogue, Payment and Cart, the MAD-VAE prototype integrated with LR-SemiVAE and RT-SemiVAE has higher $F1-score$ than the Prometheus system, confirming that MAD-VAE has better anomaly detection performance.

4.2.3 Performance evaluation of anomaly location

For the root cause entity, RT-SemiVAE provides an indicator-level fine-grained positioning of the anomaly cause and first calculates the reconstruction probability score $r_i^{(t)}$(where $i \in 1, 2, \cdots, M$) of each indicator of the abnormal entity $x^{(t)}$, and then takes them as the contribution of each indicator to an anomaly. These indicators are sorted according to the reconstruction probability score from small to large to obtain a sorted list $sortedIndex$. Finally, we use RT_t to represent the real indicators that causes $x^{(t)}$ to be an anomaly at time t.

Because there is no definite standard in indicator-level anomaly positioning to evaluate the performance of the MAD-VAE prototype, and Prometheus does not currently support this positioning method. Therefore, affected by top-K hit rate in study [32]. we propose a new quantitative method to evaluate the accuracy of MAD-VAE anomaly location, as shown in (10) below:

$$HitRate@K\% = \frac{Hit@floor(K\% * |RT_t|)}{|RT_t|} \tag{10}$$

where $|RT_t|$ is the number of indicators contained in RT_t, K may be 100 or 150, $floor(K\% * |RT_t|)$ is the downward integration of $K\% * |RT_t|$, and $Hit@K\%$ is the number

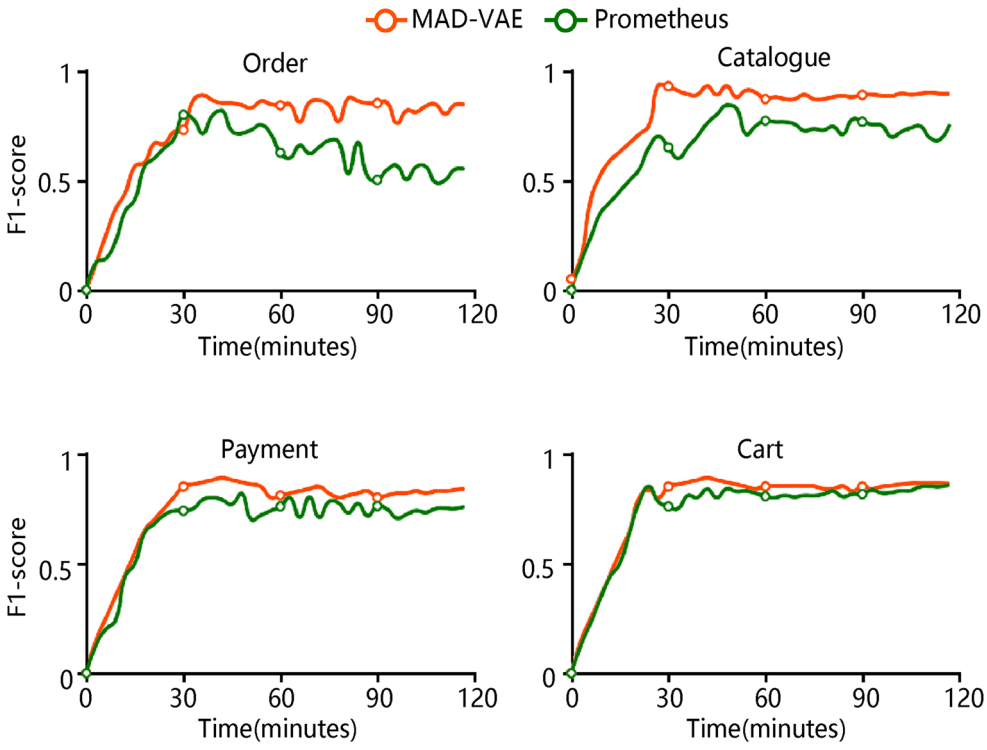


Fig. 17 Comparison of anomaly detection performance on different service entities

of overlapping indicators of $floor(K\% * |RT_t|)$ in the ordered indicator list $sortedIndex$ obtained by the MAD-VAE system and the real indicator RT_t that causes the $x^{(t)}$ anomaly.

The Sock Shop simulation data set and Yahoo data set were used to evaluate the performance of anomaly location with or without abnormal data involved in training, and the accuracy of anomaly location was compared with the MAD-GAN, VAE M2, LSTM-VAE and VAE-GAN algorithms. The results are shown in Fig. 18, which demonstrate that the performance of MAD-VAE is better than the others, and no matter which data set is used for the experiment, whether or not the model is trained with a data set with anomalies has little effect on the $HitRate@K\%$ of MAD-VAE. However, the $HitRate@K\%$ of the other baseline algorithms are lower and vary widely, which verifies that MAD-VAE achieves better performance in indicator-level anomaly localization.

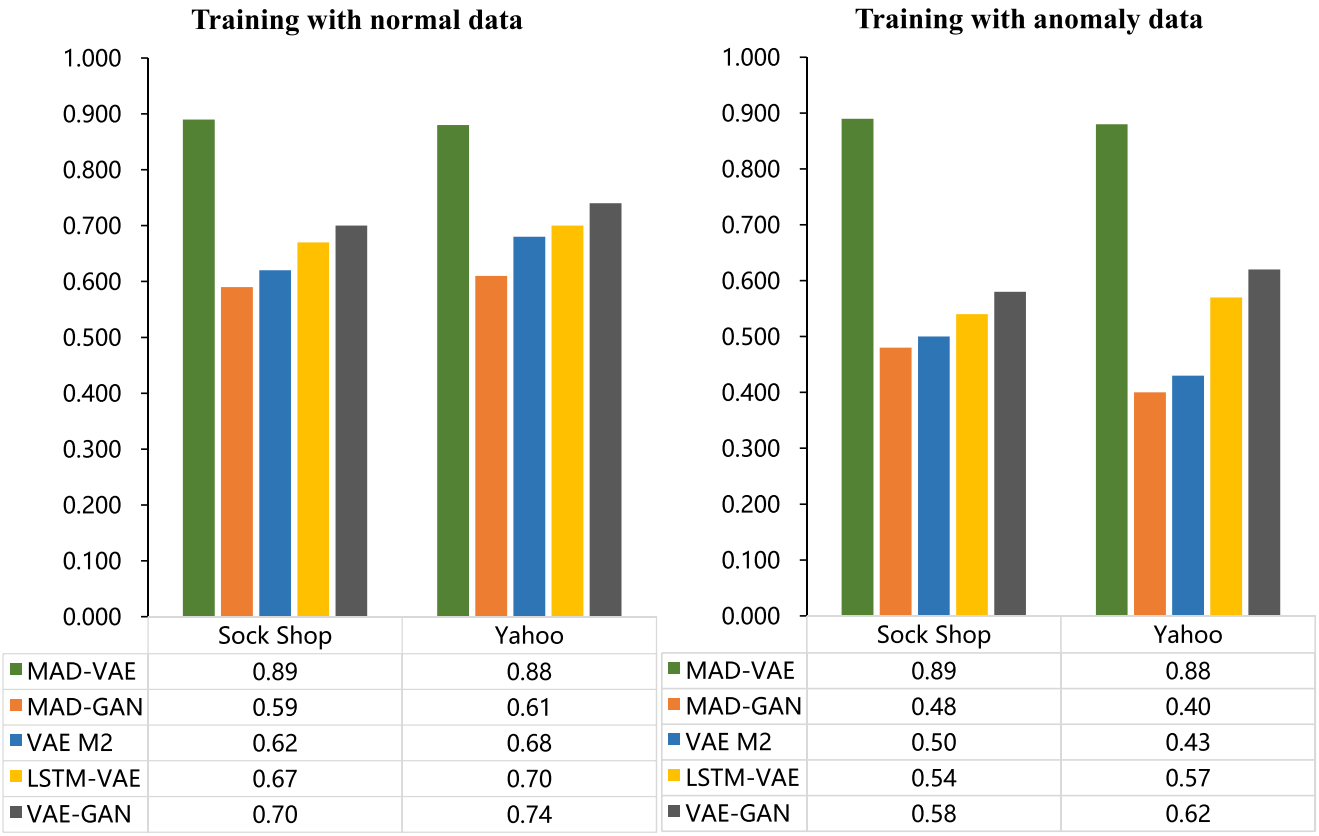
4.3 Summary of the experiments

In the algorithm comparison experiment, LR-SemiVAE and RT-SemiVAE achieved marked improvements in anomaly detection performance compared with the five baseline algorithms, where the $F1-score$ of the proposed methods are the highest, and RT-SemiVAE performed better than LR-SemiVAE in learning long-sequence time dependence and model training speed. In the prototype comparison experiment, compared with the popular cluster monitoring system, MAD-VAE achieved faster speeds and better performance in entity-oriented multivariate time series anomaly detection and indicator-level positioning, which verified that the proposed large-scale cluster AIOps scheme can improve system reliability and achieve high service availability.

5 Conclusion

In this study, to improve the performance and efficiency of multivariate time series anomaly detection for AIOps and enhance the reliability of large-scale cloud service systems, we studied three aspects and proposed two anomaly detection strategies:

1) An LR-SemiVAE anomaly detection stred. First, this strategy uses the VAE to perform feature dimension reduction and reconstruction of multivariate KPIs and judges whether the entity is abnormal by calculating the reconstruction probability score. Second, by introducing an LSTM network into the encoder and decoder of



	Sock Shop	Yahoo
MAD-VAE	0.89	0.88
MAD-GAN	0.59	0.61
VAE M2	0.62	0.68
LSTM-VAE	0.67	0.70
VAE-GAN	0.70	0.74

	Sock Shop	Yahoo
MAD-VAE	0.89	0.88
MAD-GAN	0.48	0.40
VAE M2	0.50	0.43
LSTM-VAE	0.54	0.57
VAE-GAN	0.58	0.62

Fig. 18 Comparison of $HitRate@K\%$ values with different training data

the VAE, the model can fully learn the time dependence of multivariate KPIs. Finally, LR-SemiVAE predicts the data labels by introducing a classifier to reduce the dependence on the original label data during model training.

2) A new $ELBO$ loss function calculation method is proposed, which makes LR-SemiVAE pay attention to the normal pattern and ignore the abnormal pattern during training process to reduce the time cost of removing random anomalies and noise data.
3) Based on LR-SemiVAE, we propose an RT-SemiVAE anomaly detection and location strategy for application systems with complex dependencies. This strategy learns the long-term dependence of multivariate time series by introducing a parallel multihead attention mechanism transformer, while the LSTM network is used to capture short-term dependence. Parallel computing also markedly reduces the model training time. After RT-SemiVAE detects entity anomalies, it traces the root cause entities according to the obtained service dependence graph and locates the anomaly causes at the indicator level.

We verified the strategies using public data sets and constructed the system prototype MAD-VAE. Experimental results show that compared with the existing baseline methods, the LR-SemiVAE and RT-SemiVAE strategies can detect anomalies more quickly and accurately and perform fine-grained and accurate localization of the root causes of anomalies.

In future work, automatically calculating and adjusting the threshold according to the trend of collected data should be further investigated in detail. The types of anomalies can also be investigated to assess the impact of different types of anomalies and automatically determine what type of anomaly self-healing scheme is used and implemented according to the evaluation results. We believe it is important to study how to use multivariate KPI data and log data of cluster systems concurrently to achieve efficient and accurate anomaly detection and location with graph neural networks (GNNs) [33].

Acknowledgements This work is supported in part by the National Key Research and Development Project of China under Grant 2017YFC1602005 and 2018YFB1404404, the Natural Science Foundation of China under Grant 62162003 and 61762008, and the Innovation Project of Guangxi Graduate Education under Grant YCSW2022075.

Author Contributions Ningjiang Chen: Conceptualization, Methodology, Resources, Supervision, Funding acquisition. Huan Tu: Software, Investigation, Resources, Writing - review & editing, Visualization. Xiaoyan Duan: Validation, Formal analysis, Writing original draft. Liangqing Hu: Data collection and curation. Chengxiang Guo: Data processing and analysis.

Declarations

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

References

1. Borghesi A et al (2019) A semisupervised autoencoder-based approach for anomaly detection in high performance computing systems. Eng Appl Artif Intell 85:634–644
2. Notaro P, Cardoso J, Gerndt M (2021) A survey of AIOps methods for failure management. ACM Trans on Intell Sys and Tech (TIST) 12.6:1–45
3. He S et al (2021) A survey on automated log analysis for reliability engineering. ACM Comp Surveys (CSUR) 54.6:1–37
4. Yadav RB, Kumar PS, Dhavale SV (2020) A survey on log anomaly detection using deep learning. 2020 8th International Conference on Reliability, Infocom Technologies and Optimi zation (Trends and Future directions)(ICRITO). IEEE
5. Blázquez-García A et al (2021) A review on outlier/anomaly detection in time series data. ACM Comp Surveys (CSUR) 54.3:1–33
6. Kingma DP, Welling M (2014) Auto-encoding variational bayes. In: ICLR
7. Goodfellow I et al (2014) Generative adversarial nets. Advances in neural information processing systems. vol 27
8. Hundman K et al (2018) Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding. Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining
9. Li D et al (2019) MAD-GAN: multivariate anomaly detection for time series data with generative adversarial networks. International conference on artificial neural networks. Springer, Cham
10. Park D, Hoshi Y, Kemp CC (2018) A multimodal anomaly detector for robot-assisted feeding using an lstm-based variational autoencoder. IEEE Robotics and Automation Lett 3.3:1544–1551
11. Lin S et al (2020) Anomaly detection for time series using vae-lstm hybrid model. ICASSP 2020-2020 IEEE international conference on acoustics speech and signal processing. (ICASSP) IEEE
12. Niu Z, Yu K, Wu X (2020) LSTM-Based VAE-GAN for time-series anomaly detection. Sensors 20.13:3738
13. Razavi-Far R et al (2018) Information fusion and semi-supervised deep learning scheme for diagnosing gear faults in induction machine systems. IEEE Trans on Industrial Elect 66.8:6331–6342
14. Xu Haowen et al (2018) Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications. Proceedings of the 2018 world wide web conference
15. Lindemann B‘ et al (2021) A survey on anomaly detection for technical systems using LSTM networks. Comp in Industry 131:103498
16. Ergen T, Kozat SS (2019) Unsupervised anomaly detection with LSTM neural networks. IEEE Trans on Neural Networks and Learning Sys 31.8:3127–3141
17. Zhou X et al (2020) Variational LSTM enhanced anomaly detection for industrial big data. IEEE Trans on Industrial Informatics 17.5:3469–3477
18. Huang F et al (2018) Multimodal network embedding via attention based multi-view variational autoencoder. Proceedings of the 2018 ACM on international conference on multimedia retrieval
19. Lin S et al (2020) Anomaly detection for time series using vae-lstm hybrid model. ICASSP 2020-2020 IEEE international

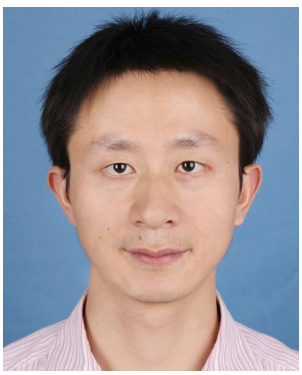
conferenc on acoustics, speech and signal processing (ICASSP). IEEE
20. Maleki S, Maleki S, Jennings NR (2021) Unsupervised anomaly detection with LSTM autoencoders using statistical data-filtering. Applied Soft Computing 108:107443
21. Geiger A et al (2020) TadGAN: time series anomaly detection using generative adversarial networks. 2020 IEEE international conference on big data (Big Data). IEEE
22. Bashar MA, Nayak R (2020) TANoGAN: time series anomaly detection with generative adversarial networks. 2020 IEEE symposium series on computational intelligence (SSCI). IEEE
23. Vaswani A et al (2017) Attention is all you need. Advances in neural information processing systems. vol 30
24. Phongwattana T, Chan JH (2019) Development of biomedical corpus enlargement platform using BERT for bio-entity recognition. International conference on neural information processing. Springer Cham
25. He J et al (2019) HSI-BERT: hyperspectral image classification using the bidirectional encoder representation from transformers. IEEE Trans on Geoscience and Remote Sensing 58.1:165–178
26. Ziyu Z, Wang Q (2019) R-transformer network based on position and self-attention mechanism for aspect-level sentiment classification. IEEE Access 7:127754–127764
27. Bian J et al (2019) A novel and efficient CVAE-GAN-based approach with informative manifold for semi-supervised anomaly detection. IEEE Access 7:88903–88916
28. Das A et al (2020) An End-to-End Approach for Benchmarking Time-Series Models Using Autoencoders. Proceedings of the Global AI Congress 2019. Springer, Singapore
29. Zhang S et al (2020) Semi-supervised bearing fault diagnosis and classification using variational autoencoder-based deep generative models. IEEE Sensors J 21.5:6476–6486
30. Song M, Zhang C, Haihong E (2018) An auto scaling system for API gateway based on Kubernetes. 2018 IEEE 9th international conference on software engineering and service science (ICSESS). IEEE
31. Chang C-C et al (2017) A kubernetes-based monitoring platform for dynamic cloud resource provisioning. GLOBECOM 2017-2017 IEEE global communications conference. IEEE
32. Lee J-W et al (2019) Collaborative distillation for top-N recommendation. 2019 IEEE international conference on data mining (ICDM). IEEE
33. Wu Z et al (2020) A comprehensive survey on graph neural networks. IEEE Trans on Neural Networks and Learning Systems 32.1:4–24

Ningjiang Chen (Member, IEEE), received the Ph.D. degree from the Institute of Software, Chinese Academy of Sciences, in 2006. He is currently a Professor at Guangxi University. His research interests include intelligent software engineering, big data, and cloud computing, etc. He is a member of ACM.

Huan Tu was born in China, in 1998. He received the bachelor's degree from the Guangxi University of E-commerce, in 2020. He is currently pursuing the master's degree with the Department of Computer Technology, Guangxi University, China. His current research interests include software engineering, big data, and cloud computing.

Xiaoyan Duan was born in China, in 1995. She received the master's degree from the Guangxi University of Software Engineering, in 2020. She has obtained several papers, software copyrights and patents in the field of intelligent software and cloud computing. Her current research interests include computer network, big data, and cloud computing.

Liangqing Hu Ph.D. Candidate in Computer Science and Technology. His main research interests include transfer learning, few-shot learning, machine learning.



Chengxiang Guo was born in China, in 1978. He received the master's degree from the Guangxi Normal University of computer application technology, in 2011. He is currently a Senior Engineer at Guangxi University of Chinese Medicine. Her research interests include image processing, big data, artificial intelligence.

Affiliations

Ningjiang Chen[1,2,3] · Huan Tu[1] · Xiaoyan Duan[1] · Liangqing Hu[1] · Chengxiang Guo[4]

Huan Tu
th_1998@163.com

Xiaoyan Duan
492089303@qq.com

Liangqing Hu
490606359@qq.com

Chengxiang Guo
guocx2019@gxtcmu.edu.cn

1 College of Computer and Electronic Information, Guangxi University, 530004, Nanning, Guangxi, China

2 Guangxi Colleges and Universities Key Laboratory of Parallel and Distributed Computing, 530000, Nanning, Guangxi, China

3 Guangxi Key Laboratory of Multimedia Communications and Network Technology, 530000, Nanning, Guangxi, China

4 Guangxi University of Chinese Medicine, 530200, Nanning, Guangxi, China