Hindawi
Mobile Information Systems
Volume 2022, Article ID 8378187, 14 pages
https://doi.org/10.1155/2022/8378187

Research Article

Anomaly Detection in QAR Data Using VAE-LSTM with Multihead Self-Attention Mechanism

Chuitian Rong,[1] **Shuxin OuYang**,[1] **and Huabo Sun**[2]

[1]*School of Computer Science and Technology, Tiangong University, Tianjin 300384, China*
[2]*Institute of Aviation Safety, China Academy of Civil Aviation Science and Technology, Beijing 100028, China*

Correspondence should be addressed to Shuxin OuYang; 2031081023@tiangong.edu.cn

Received 22 June 2022; Accepted 8 September 2022; Published 30 September 2022

Academic Editor: Chin-Ling Chen

With the rapid development of the aviation industry, it is particularly important to ensure the safe flight of aircraft. How to find potential hazards in the process of aircraft flight has always been one of the important topics of civil aviation research. At present, the Quick Access Recorder (QAR) is the most widely used equipment to store the data recorded on aircraft. QAR data contain a lot of valuable and unexplored information, which records the true status of the aircraft in detail. Therefore, finding abnormal data from QAR data lays an important foundation for obtaining the cause of abnormality and providing a guarantee for flight. In this paper, in order to discover the abnormal information in the QAR data, we applied a VAE-LSTM model with a multihead self-attention mechanism. Compared to the VAE and LSTM models alone, our model performs much better in anomaly detection and prediction, detecting all types of anomalies. We conducted extensive experiments on real-world QAR data sets to prove the efficiency and accuracy of our proposed neural network model. The experimental results proved that our proposed model can outperform state-of-the-art models under different experimental settings.

1. Introduction

With the continuous growth of civil aviation passenger traffic, aviation safety has become a significant issue in the world. Safety is the prerequisite for the steady development of all industries and the basis for the survival of the transportation sector. Ensuring aviation safety has always been a major challenge for aviation activities [1].

Flight Operational Quality Assurance (FOQA) is an important scientific method of aviation safety management, which is used to monitor the data generated by aircraft. Over the last few decades, with improved sensing capabilities, there are different recorders that have been installed on aircraft to monitor the aircraft systems and flight crew performance. In these recorders, the Quick Access Recorder (QAR) is easier to install and configure compared to the Flight Data Recorder (FDR) and Cockpit Voice Recorder (CVR). QAR is a flash recorder for aircraft data acquisition systems and is also a key data source for airlines to evaluate flight quality and aircraft engine operations [2]. It covers most of the parameters of aircraft flight, including aircraft attitude parameters and engine-related data. By analysing a number of flight parameters recorded by QAR, the anomalies can be detected to avoid safety hazards and improve flight quality.

Nowadays, aircraft failure detection and early warning based on QAR data have become one of the important fields of civil aviation scientific research. However, there are many factors that can affect the quality of QAR data, such as working environment, signal transmission, data precision, and data decode computation Therefore, the original QAR data contain many anomalies and cannot be used directly without processing. In order to improve the quality of QAR data, it is necessary to perform anomaly detection on QAR data. Anomaly detection from QAR data is also one of the important strategies of FOQA. Finding anomalies from QAR data in time can prevent many unnecessary losses. To ensure the safe flight of the aircraft, it calls for an efficient and accurate anomaly detection method using advanced techniques.

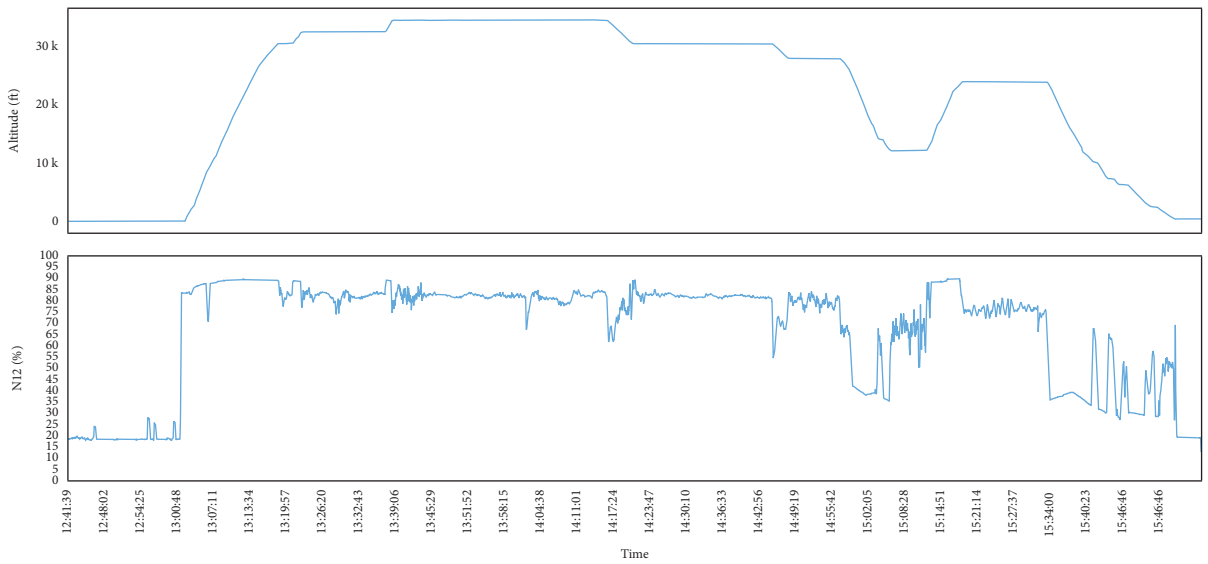

Figure 1: Example of QAR data.

QAR data records hundreds of parameters in each flight, including $N11$ and $N12$ engine thrust parameters, altitude, and vertical acceleration. It is one type of multivariate time series data. It also has the characteristics of a large amount of data, strong temporal structure, and regularity of change trends. Compared with classical time series data, QAR data have some peculiar features. The classical time series data are collected from stationary sensors. The QAR data are collected during the flight, which is divided into multiple flight phases. In different phases, the flight aircraft is working at different status and environments. Thus, the recorded parameters in QAR data present different distributions. Figure 1 takes the altitude parameters and the thrust parameters of the $N12$ engine in the QAR as an example. In actual flight, QAR data record the whole flight of an aircraft from take-off to landing, including *TAXI OUT, TAKE OFF, 2 SEGMENT, INI. CLIMB, CLIMB, CRUISE, DESCENT, APPROACH, FINAL, LANDING, and TAXI IN*, 11 phases in total, as shown in Figure 1.

In recent years, many scholars have proposed many methods for anomaly detection of time series, among which deep learning methods are popular increasingly among scholars. Due to the process of forecasting and anomaly detection with a large amount of time series data, it is unrealistic to label a large amount of data for training a model. Therefore, unsupervised anomaly detection is the preferred solution for most scholars. Anomalies can be divided into three types according to different manifestations, namely, point anomalies, collective anomalies, and background anomalies. Point anomalies are the easiest to find and can usually be marked by simple threshold or clustering methods. In contrast, collective and background anomalies are the most common in life and deserve more in-depth study. Dealing with background anomalies usually takes into account the relationship between adjacent data, and the use of models based on predictive methods is very effective for detecting such anomalies. For example, Ergen and Kozat [3] used algorithms based on long short-term memory (LSTM) neural networks to identify anomalies by calculating the difference between predicted and actual values. Collective anomalies are usually subsequences or anomalies in the entire sequence. The first step in detection is usually to divide the time series into equal-sized windows and treat the extracted subsequences as the entire sequence. For example, methods based on auto-encoder (AE) [4] and variational auto-encoder (VAE) [5], which utilize reconstruction differences for anomaly detection, have been shown to be effective.

The separate VAE model only considers the time dependence within the window and cannot analyze the information outside the window. This paper proposes a VAE-LSTM hybrid deep model based on a multihead self-attention mechanism, which integrates VAE and LSTM as a whole for unsupervised anomaly detection. Instead of directly inputting the raw data into the LSTM model like other methods, we pretrain the VAE model first and then use the low-dimensional feature vector generated by the encoder as the input of the LSTM model. Using the VAE model to effectively capture the contextual information in the window enables the LSTM model to learn longer correlations in time series. First, after pretraining the VAE model, the encoder is used to divide the QAR data into windows of a specific shape to extract the features of the recorded parameters in the QAR data, and the generated low-dimensional feature vector is used as the input of the LSTM model. Next, we use the LSTM model to train the data for the memory function of the time series. We also improve the LSTM model by incorporating a multihead self-attention mechanism, which is derived from the Transformer model [6]. Attention mechanisms are usually used in related fields such as text classification and text translation. In this paper, we apply it in time series anomaly detection. The self-attention mechanism can adjust the weight of the data, which is equivalent to a feature extraction of the data itself, and it is easier to capture

long-distance interdependent features. The multihead self-attention mechanism operates multiple self-attention mechanisms in parallel, reducing the amount of computation by reducing the dimension. Finally, the feature vector generated by the multihead self-attention mechanism module is reconstructed by the decoder for anomaly detection. We utilize a multihead self-attention mechanism for deep feature extraction. Because compared with traditional deep learning methods, it can explore hidden features without relying on complex neural network structures and have higher efficiency and performance than them. It makes it easier to capture long-distance interdependent features. We believe that combining the multihead self-attention mechanism with the LSTM model can better focus on the long-term dependencies of time series. In this way, our model can effectively detect both short-term anomalies and long-term anomalies.

In summary, the main contributions of this work are as follows:

(i) We first pretrain the VAE model and optimize the model by maximizing the ELBO loss for feature learning. We propose a novel anomaly detection model for QAR data.

(ii) We improve the LSTM model by incorporating a multihead self-attention mechanism to capture long-term correlations in QAR data. It is able to detect all types of exceptions.

(iii) In order to further improve the classification accuracy, we adopt a threshold selection method that maximizes the $F1$ metric, which effectively reduces false positives caused by improper threshold selection.

(iv) We conducted extensive experiments on the real QAR dataset to evaluate our model and compared it with other deep learning methods. Experiments show that our model has a significant improvement over other methods.

2. Related Works

QAR data are multivariate time series data with unique characteristics compared to classical time series data. Few works focus on the anomaly detection of QAR data. In this section, we first discuss existing state-of-the-art methods in the field of anomaly detection and analyze their strengths and weaknesses in order to justify our proposed method.

Anomaly detection of time series has always been a complex and challenging task in many disciplines and has been widely studied by many scholars. In anomaly detection, temporal continuity is important. Outliers are often those that are defined as unusual due to a lack of continuity in their short or long history. Therefore, anomalies in time series can be divided into two categories: short-term anomalies and long-term anomalies. Short-term anomalies occur when there are sudden changes in series values or short time intervals in the time series. The long-term anomaly is the entire time series or a subsequence that is identified as anomalous. In the past, the field of anomaly detection has generated a large amount of literature. We can roughly classify their proposed methods into three categories: statistics-based methods, classical machine learning-based methods, and deep learning-based methods [7].

2.1. Statistical Methods. The most common methods are autoregressive moving average (ARMA) and one of its generalizations, differential autoregressive moving average (ARIMA) [8]. They are one of the classic prediction-based anomaly detection models and are suitable for univariate time series. The ARIMA model uses previous data to fit a linear equation for prediction, describes the relationship between current and historical values, and uses its own historical data to predict new data. It requires that the sequence be stationary, and for nonstationary sequences, it needs to be stationary by difference. Ottosen and Kumar [9] used the ARIMA anomaly detection technique to detect short-term anomalies in low-cost air quality datasets by calculating prediction errors based on the absolute value of the residuals. However, the disadvantage of this method is that it can only predict phenomena related to the previous data, and the number of autoregressions and the parameters of prediction error need to be selected appropriately.

2.2. Machine Learning Methods. Common machine learning algorithms include clustering methods such as K-means clustering [10]. The K-means algorithm is the basic and most widely used partitioning algorithm in clustering methods. The sample data are clustered by the specified number of categories K, and the corresponding cluster centroids are used to detect anomalies in the monitoring data. Li et al. [11] proposed a cluster-based algorithm to detect excessive QAR events. It converts each flight data into a high-dimensional vector and uses the DBSCAN algorithm to cluster the matrix row vectors. The purpose is to identify exceptions without knowing the normative standard. Zhao et al. [12] proposed an algorithm based on a Gaussian mixture model (GMM) that incrementally updates the clusters according to the data instead of reclustering and adapts to the new data through an expectation-maximization algorithm to handle dynamically changing data in flight data. Zeng et al. [13] used a density-based DBSCAN clustering method to detect aircraft onboard and controller data that deviate from the normal range. Edward Smart et al. [14] proposed a two-stage approach based on a support vector machine (SVM) classifier to detect anomalies in the descent stage of a specific flight. The first stage quantifies anomalies at specific altitudes during the flight, and the second stage ranks all flights to identify the most likely anomalies. Although the above algorithms can detect abnormal flights from QAR data, they do not take into account the temporal patterns between the data and do not better explain why the abnormality occurs.

2.3. Deep Learning Methods. Compared with the above two methods, deep learning-based anomaly detection models can capture more complex hidden features and temporal

correlations in time series, so they have received extensive attention in recent years. Broadly speaking, they can be divided into two categories: predictive models and generative models. Predictive models detect anomalies based on the error of the prediction as an anomaly score. In particular, convolutional neural network (CNN), recurrent neural network (RNN), and an improved model based on it, long short-term memory network LSTM [15], have achieved remarkable results. They all have a powerful ability to learn from data. In addition, LSTM networks can model longer data; it has control structures (gates) to regulate stored memory and learn and capture normal behavior. When encountering data that deviate significantly from normal data, it predicts a large error to indicate anomalies. Hundman et al. [16] used an LSTM model for the prediction of spacecraft telemetry data and used dynamic thresholding of errors to identify anomalies. Khorram et al. [17] combined CNN and LSTM as a novel convolutional long-short-term memory recurrent neural network for fault detection, achieving high generalization accuracy and resistance to overfitting. However, such a separate prediction model is not only computationally expensive but also may lead to large deviations in the prediction results due to some uncertain factors. LSTM models are also very sensitive to the choice of parameters. As a result, many advanced generative models have emerged, including variational autoencoder (VAE) [18] and generative adversarial networks (GAN) [19]. At their core, they learn representations of normal patterns. Kishore et al. [20] proposed a deep autoencoder (DAE) applied to the raw time series data of multiple aircraft sensors and used the error of AE reconstruction to determine whether the data were abnormal. Combining convolutional neural network (CNN) with VAE, Memarzadeh et al. [21] developed a convolutional variational autoencoder (CVAE) applied to the data of abnormal commercial flight departures. Wang et al. [22] proposed a sequential parameter attention-based convolutional autoencoder (SPA-CAE) model for feature extraction from Changshui Airport in Kunming QAR data. Provotar et al. [23] used LSTM layers in an autoencoder framework. Considering that, compared with normal data, abnormal data are difficult to be represented by low-dimensional feature vectors. By inputting the data into the LSTM autoencoder, the error of the AE reconstruction is used to judge whether the data are abnormal. These generative models hold great promise in the field of anomaly detection. However, these reconstruction-based models are difficult to capture long-range temporal dependencies and cannot explicitly address potential interactions between features. On the other hand, simply adding a network such as LSTM to a feedforward layer in AE or VAE does not perform detection well.

In summary, the information inside the window after dividing the window and the correlation between the window and the remaining time series are essential in anomaly detection. Although many approaches have been proposed, it is often impossible to achieve both. The correlation between windows is ignored and only one type of anomaly is detected. Based on these reasons, we propose a new VAE-LSTM hybrid deep model based on a multihead self-attention mechanism, which can effectively identify multiple types of anomalies without the limitation of window size.

3. Model

In this section, we introduce the overall workflow and internal structure of the VAE-based MHSA-LSTM hybrid model, as shown in Figure 2. We will introduce our model training process in an unsupervised way and explain the anomaly detection process on QAR data.

3.1. Problem Definition. A univariate time series is an ordered sequence of n real-valued variables arranged in chronological order. It can be formalized as $T = \{x_1, x_2, \ldots, x_n\}, T \in R$, where n is the length of the time series. Anomalies are observations or sequences of observations that deviate significantly from the general distribution of the data. In this paper, our goal is to discover outliers in QAR data through anomaly detection. Our method is divided into two parts: model training and anomaly detection. T as the training input can get a reconstructed sample T', calculate the anomaly score between T' and T, and compare it with the threshold to get the anomaly. Given a binary variable $y \in \{0, 1\}$, $y_t = 1$ is used to indicate that an anomaly occurred in the window of time t, and $y_t = 0$, no exception occurred.

3.2. Data Preprocessing. Data preprocessing is essential when building neural network models and can often determine the results of model training. First, we need to divide the given time series into a training set and a test set. A continuous data segment that does not contain anomalies is used as training data, and the rest with abnormal data is used as test data. Then, to improve the robustness of the model, we need to standardize the training set and test set. We first standardize the training set and then use the standardized parameters (mean and variance) of the training set to standardize the test set. The data standardization formula can be expressed as the following equation:

$$x' = \frac{(x - \mu)}{\sigma}, \tag{1}$$

where μ and σ are, respectively, represented as the mean and variance of the training set.

3.3. Training Model

3.3.1. Pretraining Using VAE Model. The VAE model is a typical generative model, which consists of two parts: an encoder and a decoder. First, we preprocess the input data X and send it to the encoder, which can encode higher data dimensions into a potential representation space Z, which is random and low-dimensional. The mean and variance of the output generate the corresponding latent variable z that satisfies the unit Gaussian distribution, so we can express the encoder as $q^{\phi}(z \mid X)$, and the parameter ϕ represents the mapping of the network from X to z. The other part of the

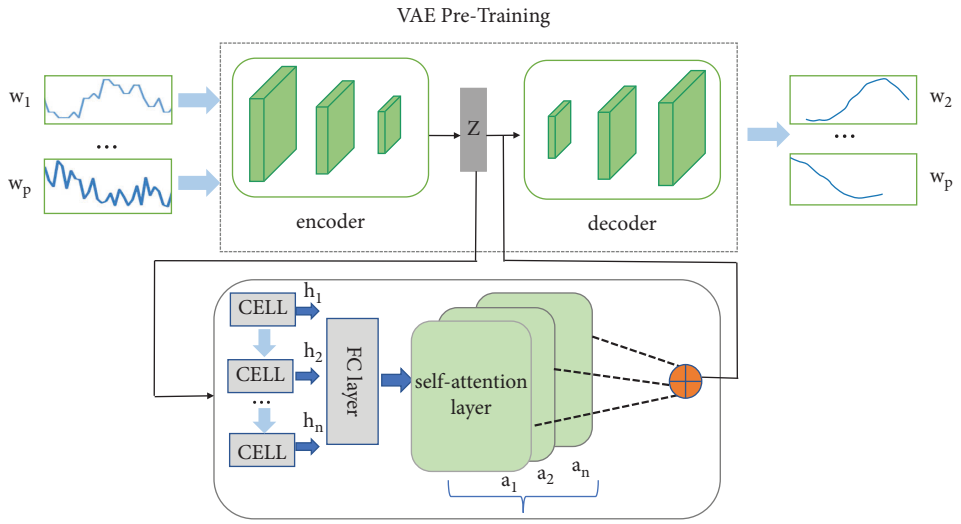

Figure 2: The workflow of the anomaly detection model.

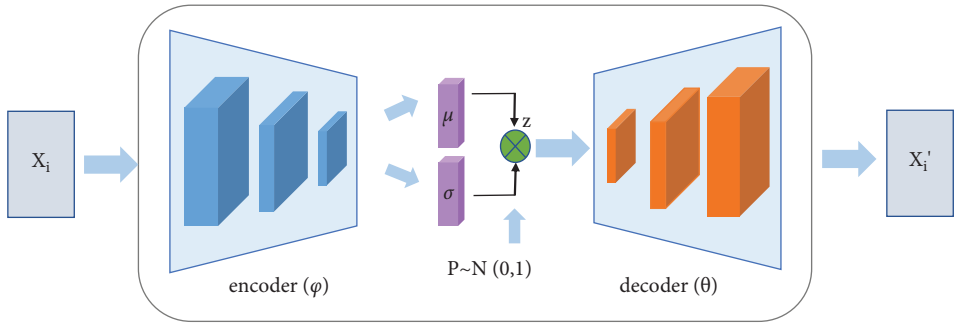

Figure 3: Architecture of variational autoencoder.

decoder of the VAE model can decode the latent variable z into the generated data x' which is similar to the real data and obeys the normal distribution with mean μ and variance σ, that is, $p \sim N(\mu, \sigma)$. Thus, we can express the decoder as $p^{\theta}(X \mid z)$, and the parameter θ represents the reconstruction of the network from z to X. Figure 3 shows the structure of the VAE network model.

In order to train the VAE model, we convert the training data into a local window as the input of the model, extract the features through the encoder, compress it into the latent space, and then reconstruct it. Given a time series $T = \{x_1, x_2 \ldots x_n\}$, where $x_i \in R$, each data point is the result of measurement at a characteristic time. To improve the accuracy of the model, we need to divide the entire time series into multiple subsequences, which are represented by time windows. We define w_t, a time window of length m at a given time t: $w_t = \{x_{t-m+1}, \ldots, x_{t-1}, x_t\}$. Because we use m data to predict the output, a total of $n - (m + 1)$ windows can be generated for training the VAE model. In this way, the time series T can be represented by the training input window sequence W: $W = \{W_1, W_2, \ldots, W_{n-m+1}\}$. After training, the model finally outputs the reconstructed window w_t' after the reconstruction of the window w_t through the decoder.

The loss function is the most basic and critical element used to measure the pros and cons of a model. The loss function of the VAE model is used to measure the information loss in the reconstruction process, and it is composed of the sum of the reconstruction error and the regularization term. Our VAE is trained with a loss function as shown in the following equation:

$$L(\theta, \phi) = -E_{z \sim q^{\phi}(z \mid x)}\left[\log p^{\theta}(x, z)\right] + KL\left(q^{\phi}(z \mid x) \| p(z)\right). \quad (2)$$

The first term is the reconstructed negative log-likelihood loss −ELBO (evidence lower bound), and the second term is the KL difference between $q^{\phi}(z \mid x)$ and $p(z)$. Our goal of training the VAE model is to minimize the sum of this reconstruction loss and KL divergence, which is equivalent to maximizing the ELBO loss to find the most suitable parameters θ and ϕ [24]. The objective function is the following equation:

$$\arg\max_{\theta, \phi} E_{z \sim q_{\phi(z \mid x)}}\left[\log p_{\theta(x \mid z)}\right] - KL\left(q_{\phi}(z \mid x) \| p_{\theta}(z)\right). \quad (3)$$

Through training, we optimized the parameters of the model while improving the loss, and the network finally converged, and a good generative model was obtained.

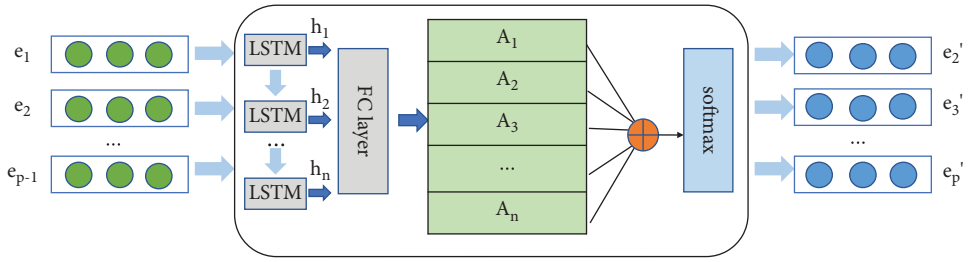

Figure 4: Architecture of the LSTM model with multihead self-attention.

3.3.2. LSTM Model Based on Multihead Self-Attention. A VAE model alone cannot achieve forecasting of time series because a VAE cannot encode or decode data outside the time window. Therefore, we use the LSTM model to act on the data after the dimension reduction of the VAE model, extract time features, and perform sequence prediction. We also introduced a multihead self-attention mechanism in the LSTM model to capture relevant information in different subspaces and highlight the importance of different features. The model structure diagram is shown in Figure 4. Below, we introduce the detailed information about the model.

(1) LSTM. The LSTM model is a type of recurrent neural network (RNN) that can solve the gradient descent or explosion problem that RNN may generate and learn and remember long-term relationships. Therefore, the LSTM network has achieved great success in time series data analysis [25]. The LSTM model is composed of LSTM memory cells. Each memory cell contains three gates with different functions, which are the input gate, output gate, and forget gate. These three gates are used to determine whether to accumulate or eliminate the information in the memory unit and to selectively retain the characteristics of the sequence. In this way, the network can determine the predicted output under this gating mechanism. Therefore, LSTM has become the basic framework for the task of processing sequential data with time information.

After pretraining the VAE model, we start to train the LSTM model. To prevent our model from overfitting, we divide the given training data into a sequence of p nonoverlapping rolling windows, which can be expressed as $W_t = [w_{t-(p-1)*m}, w_{t-(p-2)*m}, \ldots, w_t]$. Then, the window sequence W_t is encoded into a lower dimension by the encoder in the pretrained VAE model, and the output embedding can be expressed as $E_t = [e_1, e_2, \ldots e_p]$, where e_i represents the embedding of the i-th window in W_t. We train the encoder's output E_t as the input of the LSTM model and predict the next sequence e_2' based on the embedding e_1 of each window. Specifically, the LSTM model has n memory units, and each unit has a different set of internal weight parameters, namely, h and c. In each unit, there are two input data, respectively, the output and state $h_{(t-1)}$ and $c_{(t-1)}$ of the previous neuron and the input e_t of the current unit. Then, the hidden state of the output of the final unit can be expressed as the following equation:

$$h_t, c_t = \mathrm{LSTM}\left(e_t', h_{t-1}, c_{t-1}\right). \tag{4}$$

We express the hidden state of the embedded sequence e_t after passing through n LSTM units as the following equation:

$$H = \left(h_1, h_2, \ldots h_n\right). \tag{5}$$

(2) Multihead Self-Attention. Multihead self-attention is the core part of the transformer encoder-decoder model. It optimizes the traditional attention mechanism and greatly improves its performance. When performing feature extraction on a time series, you can focus your attention on a window sequence and assign weights to each time point of the sequence so as to determine the weight of their influence on the final output prediction results. An attention function is composed of a vector query, a key, and a value. The common attention mechanism is to make k and v equal to the input value, and q comes from the outside. After calculating the weight coefficients through the vectors q and k, the weighted summation with the vector v is performed to obtain the attention score. The self-attention mechanism obtains q, k, and v by making its own linear changes to the input value. Calculating the association between its own data is a feature extraction of the data itself. The calculation method of the self-attention mechanism is as shown in the following equation:

$$\mathrm{Attention}\left(Q, K, V\right) = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{6}$$

where Q is the query, K is the key, V is the value, and d_k is the number of hidden units of the neural network. The multihead self-attention mechanism performs separate operations on the basis of the self-attention mechanism. Each head generates three vectors Q, K, and V through linear transformation and then performs self-attention calculations. Calculating once is a head, and calculating h times is the so-called long head. Finally, each head is spliced and converted into the same dimension as the input sequence. The formula is expressed as the following equations:

$$A_i = \mathrm{self} - \mathrm{att}\left(QW_i^Q, KW_i^K, VW_i^V\right), \tag{7}$$

$$\mathrm{Multihead}\left(Q, K, V\right) = \mathrm{Concat}\left(A_1, A_2, \ldots A_n\right)W. \tag{8}$$

We deploy it after the LSTM model, because when calculating each head, the parameters W after the linear transformation of Q, K, and V are different, which needs to be learned by the model. We use W_i to represent. The

attention layer takes the entire hidden state H as input and multiplies it with the parameter, W_i, to calculate the self-attention value of each head. The calculation formulas are shown as the following equations:

$$u_i = \tanh\left(W_i H + b_i\right), \tag{9}$$

$$v = \text{Multihead}\left(u_1, u_2, \ldots u_p\right). \tag{10}$$

After p operations, we join each operation result u_i to get a feature representation v. Finally, the obtained feature representation vector is sent to the softmax layer for prediction, and the prediction results are as shown in the following equation:

$$y = \text{softmax}\left(\text{wv} + b\right), \tag{11}$$

where w and b are the weight matrix and bias of the final linear layer. Finally, we train our model by minimizing the error between the original data and the predicted data.

3.4. Anomaly Detection. Our anomaly detection method is divided into three stages: preprocessing, training, and detection. Among them, the training and detection stages share the first data preprocessing stage, and the data are standardized and divided into time windows of length m. After training, our model can be used for anomaly detection. First, we input the preprocessed test set sequence W_t into the LSTM model, which represents the pm data contained in time t. Then, we use the pretrained VAE model to reduce the dimensionality of W_t and encode W_t into a low-dimensional space by extracting features to obtain an embedding sequence E_t. The coded representation is used in the prediction stage of the LSTM model. The LSTM model predicts the next embedding e_i by learning $e_{(i-1)}$, as shown in the following equation:

$$e_i' = \text{LSTM}\left(e_{(i-1)}\right). \tag{12}$$

Finally, we use the decoder of the VAE model to perform feature restoration and reconstruct the predicted e_i' into a new window $W_{t-(p-i)*m}$, which is as shown in the following equation:

$$w_{t-(p-i)*m'} = \text{Decoder}\left(e_i'\right), \tag{13}$$

where $i \in \{2, 3, \ldots p\}$.

In the anomaly detection stage, our model will get a total of two results, which are the predicted value calculated based on the prediction model and the reconstructed value obtained based on the reconstruction model. We measure the degree of anomaly by calculating the root mean square error (RMSE) between the reconstructed window and the original window as the anomaly score of the window. The higher the abnormality score, the greater the possibility of abnormality. The formula to calculate the RMSE error is as shown in the following equation:

$$\text{score} = \sqrt{\sum_{i=2}^{p}\left(w_{t-(p-i)*m'} - w_{t-(p-i)*m}\right)^2 / p}. \tag{14}$$

Among them, $w_{t-(p-i)*m}$ is the true value and $w_{t-(p-i)*m'}$ is the reconstruction value. The calculated result is the sum of the reconstruction errors of each time step of the entire window. The sum of the errors of these data points can be used as the anomaly score of the entire window. In order to effectively detect anomalies, we also need to set a threshold θ on the anomaly score. If the anomaly score is higher than this threshold, we will regard the window sequence as a window where anomalies may occur.

For this kind of binary classification problem, it is essential to choose an appropriate threshold, which can maximize the performance of the classifier. Some commonly used methods of threshold selection include artificially setting a fixed threshold. When the reconstructed value is greater than (or less than) the fixed threshold, it is judged that the value is abnormal. There are also some models that detect anomalies through the 3-sigma method. Standard deviation is a commonly used quantitative form that reflects the degree of data dispersion, and the dispersion is the most basic and important indicator for evaluating the quality of a method. Therefore, when the outlier exceeds 3 times the standard deviation, it can be regarded as an outlier. The advantage of these methods is simplicity, but obviously, the solution of setting a fixed threshold during deployment is not enough, and it is prone to false positives and underreports, and the scene adaptability is low. In order to avoid the above situations and better illustrate our model, we use a method of maximizing the $F1$ metric to automatically select the best threshold. The $F1$-score value is the harmonic average of the precision rate and the recall rate, and the accuracy and recall rate of the model can be considered at the same time in the detection. It can be calculated by the following equation:

$$F1 = 2 * \frac{(P * R)}{(P + R)}. \tag{15}$$

In the formula, P represents the accuracy rate of the detection model, and R represents the recall rate of the detection. First, we compute the reconstruction error RMSE for each window as the anomaly score and then compute the $F1$-score for multiple thresholds using an iterative grid search between the minimum and maximum reconstruction errors. We record the selected threshold θ when the $F1$-score value is the highest and use it as the optimal threshold. Any sequence of windows above this threshold will be considered anomalous. Because the number of anomalies in QAR data is low, we mainly focus on continuous anomalies or anomalous segments. If any point in the anomaly segment is correctly detected, all points in the anomaly window are identified as true positives, and the others are considered normal.

4. Experimental Evaluations

In this section, we conduct several comparative experiments to demonstrate the effectiveness of our method from different perspectives. We first introduce the real-world QAR dataset used in the experiments, evaluate the performance of our model on the dataset, and compare it with

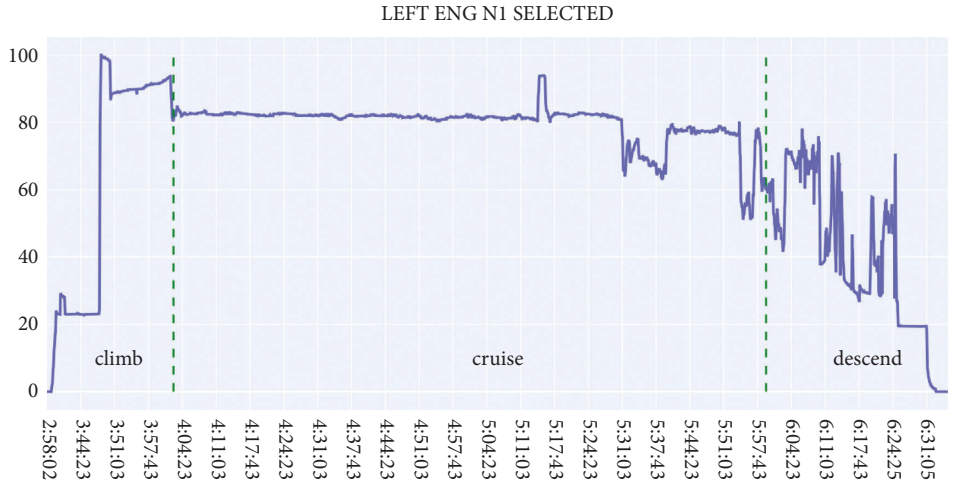

Figure 5: The $N1$ engine parameter of QAR.

Table 1: Comparison of anomaly detection performance based on precision, recall, and F1-score.

Methods	Climb			Cruise			Descent		
	P	R	$F1$	P	R	$F1$	P	R	$F1$
IF	0.5833	0.3293	0.4211	0.6333	0.7039	0.6667	0.5108	0.7055	0.5926
LSTMS	0.6944	0.9801	0.8195	0.4425	0.9440	0.6136	0.8240	0.6352	0.7769
LSTM-AE	0.8885	0.9426	0.9147	0.8768	0.9417	0.9134	0.7284	0.8534	0.7860
LSTM-VAE	0.7722	0.9443	0.8496	0.8119	0.9716	0.8961	0.8902	**1.0**	0.9419
VAE-based MHSA-LSTM	0.9145	0.9833	0.9503	0.8840	**1.0**	0.9384	0.9453	**1.0**	0.9718

other state-of-the-art methods (4.1). Second, we analyze how different parameter sizes affect the performance of the method (4.2). Then, we analyze the time performance of the model (4.3), and finally, we evaluate our algorithm (4.4) on different real-time series datasets.

4.1. The Comparisons with Different Methods

4.1.1. Datasets. Each QAR data file records the whole process of an aircraft from take-off to landing, including 11 stages, TAXI OUT, TAKE OFF, 2 SEGMENT, INI. CLIMB, CLIMB, CRUISE, DESCENT, APPROACH, FINAL, LANDING, and TAXI IN. These stages can be generally divided into three processes: the climb, cruise, and descent of the aircraft. Figure 5 shows the thrust parameters of the $N1$ engine generated by an aircraft of an airline during a voyage. It can be clearly seen that the aircraft tends to climb, stabilize, and then descend during the voyage.

Since the amplitude and speed of the data changes in different flight stages of each flight aircraft are different, we adopt a segmentation method. We divide the entire data into three segments: climate, cruise, and descend, according to the flight stage parameter FLIGHT_PHASE, and pass through each stage, respectively. The sliding window extracts local features for anomaly detection. Segmentation not only reduces the dimension of the data but also reduces the amount of computation and enhances the adaptability of the algorithm to QAR data. In this experiment, the $N1$ parameters generated by 100 normal flights of the same aircraft in the real world are selected, and each segment is connected to a file for training and anomaly detection after segmentation, and the data will have obvious circularity. There are a total of 82,606 sampling values in the climbing stage; 107,758 sampling values in the cruise stage; and 69,330 sampling values in the descending stage.

4.1.2. Experimental Setup. Our proposed method is mainly implemented by the Python programming language. It uses the well-known Tensorflow and Keras deep learning frameworks and includes multiple statistics and visualization packages, including Sckit-learn, Pandas, and Numpy. For the hyperparameters used in the model training process, we set the hidden size of the LSTM unit to 64 by default; $h_{\dim}$ for dimension of the hidden layer in the VAE model is set to 512, and $z_{\dim}$ for dimension of the latent variable Z is set to 10; the number of heads n of multihead self-attention is set to 6; and the number of samples for each training batch size is set to 64. In the training details, the learning rate of the VAE model and LSTM model is set to 0.0002; adaptive moment estimation (Adam) is used as the optimizer to optimize the gradient. The reconstruction loss of the mean square error MSE serves as the loss function of the LSTM model, while the loss function of the VAE consists of the reconstruction loss of the mean square error MSE and the Kullback-Leibler divergence loss of the difference between the target distributions. The model is trained for 50 epochs. For the other comparison models, we also use the hyperparameters described above. All models that require sliding windows are compared under the condition that the default window length is 144. When testing each model, we retained the results with the highest $F1$-score.

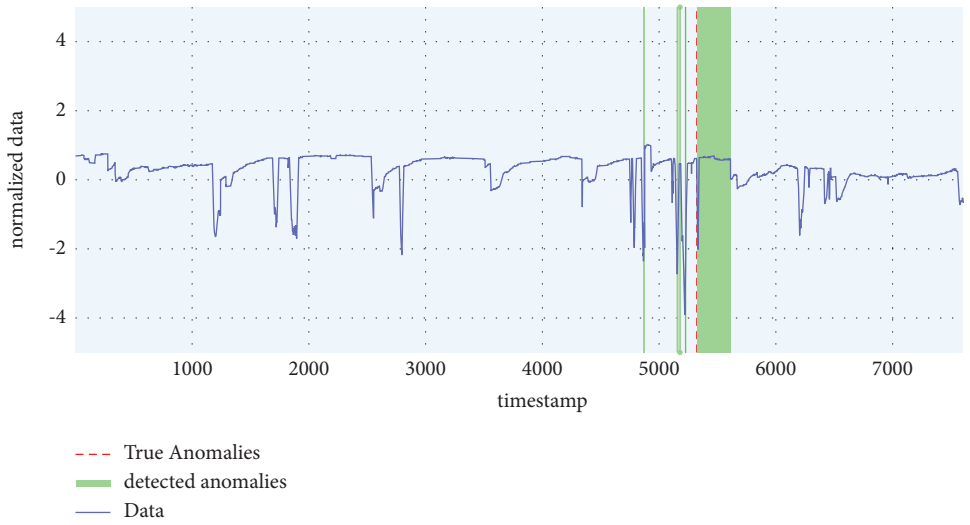

Figure 6: Anomaly detection in the climb phase dataset.

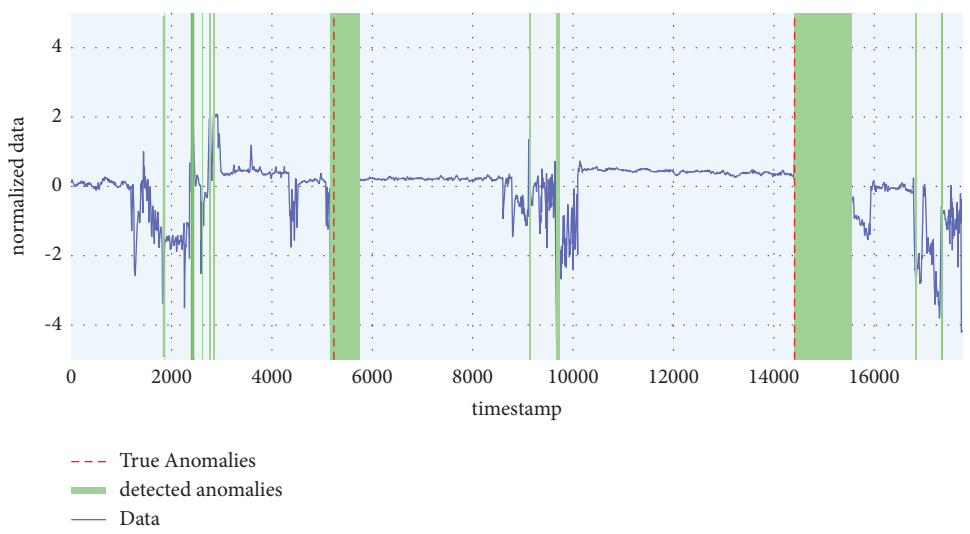

Figure 7: Anomaly detection in the cruise phase dataset.

4.1.3. Evaluation Metrics. The performance indicators used in the comparison experiments are precision, recall, and $F1$-score, which are commonly used evaluation indicators in anomaly detection. Equation (15) already gives the definition of the $F1$-score. Equations (16) and (17) give the definitions of precision and recall.

$$\text{precision} = \frac{TP}{(TP + FP)}, \tag{16}$$

$$\text{recall} = \frac{\text{TP}}{(TP + FN)}, \tag{17}$$

where TP stands for true positives, FP stands for false positives, and FN stands for false negatives. When a window is detected and marked as abnormal, where TP is the number of correctly detected abnormal points, FP is the number of normal points that are incorrectly predicted as abnormal points, and FN is the number of abnormal points that are incorrectly predicted to be normal. Accuracy is the ratio of the number of correctly predicted samples to all predicted samples of a particular class and can be used to measure the quality of model prediction. Recall is calculated as the ratio of correctly predicted samples to the total number of instances of the same type. The higher the recall, the easier it is for the model to detect anomalies. Higher recall is very important. Precision is generally paired with recall to evaluate model performance, but sometimes there are contradictions. Therefore, in order to have a more comprehensive evaluation of anomaly detection, we more comprehensively consider the $F1$-score, which is the harmonic mean of precision and recall.

4.1.4. Results. To demonstrate the overall performance of our proposed method, we compared it with four other unsupervised anomaly detection models. They are isolation forest (IF) [26], long-short-term memory (LSTMS) [16],

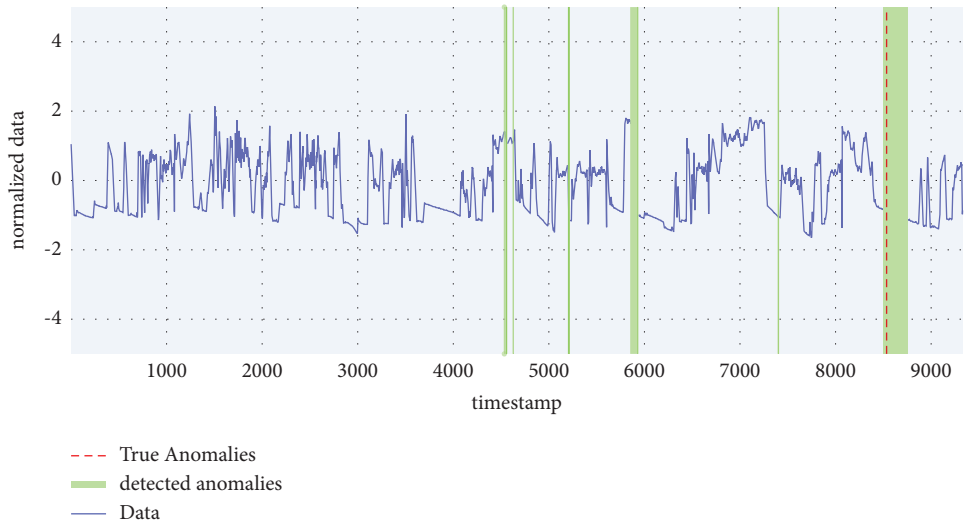

Figure 8: Anomaly detection in the descent phase dataset.

Table 2: The effect of head number.

Heads	Climb			Cruise			Descent		
	P	*R*	*F*1	*P*	*R*	*F*1	*P*	*R*	*F*1
2	0.8119	0.9964	0.8947	0.8584	0.9826	0.9238	0.8511	1.0	0.9196
4	0.8299	0.9960	0.9054	0.8957	0.9803	0.9449	0.9127	0.9987	0.9543
6	0.9145	0.9890	0.9503	0.8782	1.0	0.9356	0.9265	1.0	0.9618
8	0.8465	1.0	0.9119	0.8934	0.9763	0.9437	0.8707	0.9981	0.9309

Table 3: Training time per epoch (min).

Methods	Climb	Cruise	Descent
LSTM-AE	5.05	7.57	4.27
LSTM-VAE	2.04	1.69	2.58
VAE-based MHSA-LSTM	1.75	1.32	1.2

LSTM-AE [27], and LSTM-VAE [28]. We report for each method the results associated with the highest *F*1-score values. Table 1 details the performance results of all methods on the climb, cruise, and descent datasets. The results show that our model significantly outperforms other methods in precision, recall, and *F*1-score on all datasets, where the precision is able to improve by 0.5–0.7. It can also be observed that our model performance is well-balanced across different stages of QAR data. Figures 6–8 show the visualization results of our method for anomaly detection on the climb, cruise, and descent datasets. From these figures, we can see that our method can correctly find the time window in which abnormal events occur, which proves that our model has a high recall rate. The very few false positives plotted in the graph are because, historically, such a spike has been infrequent, so it was detected as an anomaly by our model. We may need more domain knowledge to solve this problem in the future.

The methods we compare include the machine learning method IF, the traditional predictive model LSTM, and the combination of LSTM with autoencoders and generative models. It can be observed that the IF method performs the worst. IF builds a collection of iTrees for a given dataset, and then the instances go through all iTrees. The anomaly score is the average of all path lengths. It does not observe time information. In time series, time correlation is essential. The prediction model composed of LSTMs may lead to large deviations in the results due to the uncertainty of the prediction results. Thus, the results of precision and recall are relatively low. The autoencoder reconstructs time series through an encoder-decoder framework. On this basis, LSTM is combined with the autoencoder, and the encoder and decoder of AE are composed of multiple LSTM units. The main role of AE is to reduce the dimensionality of the data, form a low-dimensional latent vector, and combine it with LSTM to capture the long-term correlation of time series. However, in contrast, as a generative model, VAE can generate new data completely different from the training data through training and satisfy the standard normal distribution. It can be seen from the experimental results that the combination of VAE and LSTM is much better than AE. Detection performance improved, but significant performance fluctuations were seen between different stage datasets. Our method adds a multihead self-attention mechanism on top of this and calculates the dependencies between long-distance windows separately through multiple heads. The weighting calculation is applied to the reconstruction of the VAE decoder. Therefore, our model captures the long-term dependencies of time series more easily than other methods. The results also show that our model achieves 100% recall on the cruise and descent datasets. This

Table 4: Statistical information of four public benchmark datasets.

Dataset	Total length	Train length	Test length	Mean	Std.	Anomaly rate (%)
KPI1	90000	75000	15000	2.3840	0.9174	5.18
KPI2	17562	10000	7562	0.1911	0.1004	0.79
NAB1	18050	15500	2550	37.4794	14.4096	0.08
NAB2	4032	3000	1032	45.1079	1.8774	0.29

Table 5: Anomaly detection performance on four public benchmark datasets.

Methods	KPI1			KPI2			NAB1			NAB2		
	P	*R*	*F*1	*P*	*R*	*F*1	*P*	*R*	*F*1	*P*	*R*	*F*1
LSTMS	0.7639	0.6544	0.7049	0.5850	0.9997	0.7382	0.4536	**1.0**	0.6241	0.8604	0.7915	0.8779
LSTM-AE	0.7261	0.8521	0.7841	0.6773	0.8230	0.7430	0.7611	0.6807	0.6870	0.7627	**0.8733**	0.8142
LSTM-VAE	0.7815	0.9545	0.8594	0.8734	0.9271	0.8995	0.7468	**1.0**	0.8550	0.9090	0.6563	0.7623
VAE-based MHSA-LSTM	**0.8221**	**1.0**	**0.9023**	**0.8786**	**1.0**	**0.9354**	**0.8731**	**1.0**	**0.9322**	**0.9547**	0.8146	**0.8791**

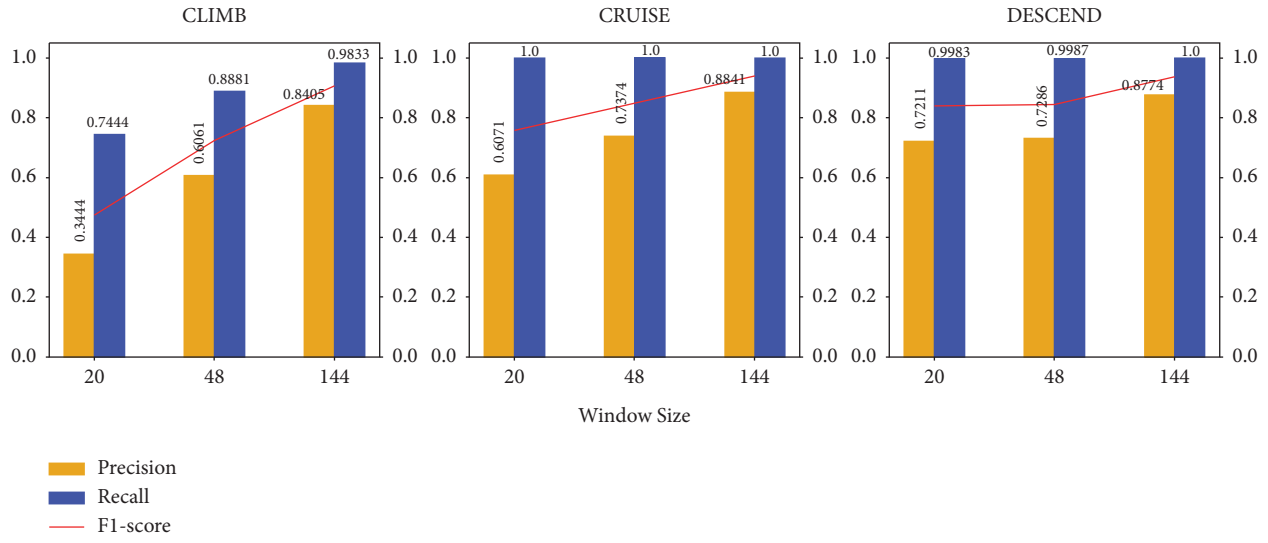

Figure 9: Comparisons with different window sizes.

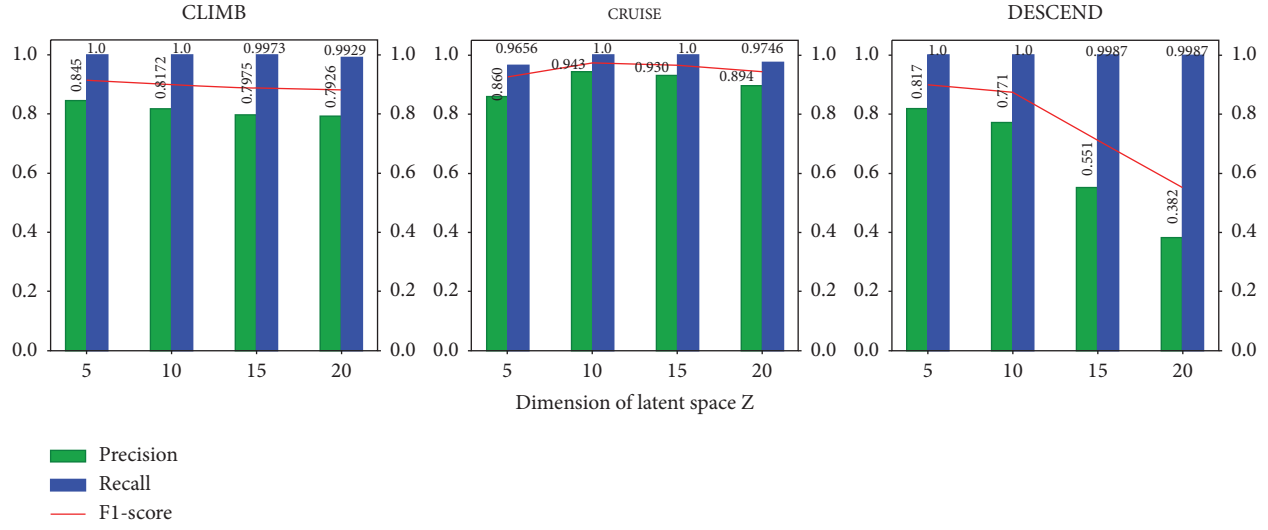

Figure 10: Comparisons with different z_dim.

shows that we have no abnormal points that are wrongly predicted to be normal and can effectively detect both short-term and long-term anomalies. Overall, our method shows better performance than other methods.

4.2. Effect of Parameters. In this section, we investigate the different effects of different parameters and factors on the method's performance, and all experiments are done using the three datasets of QAR.

4.2.1. Effect of Different Window Sizes. The first factor is the different window sizes in different datasets. The window size has an impact on the results of anomaly detection, because it not only affects the speed and efficiency of anomaly detection but also directly affects the detection accuracy. It is crucial to model the data within the window interval by choosing the appropriate window size for different datasets. We set the window size to 20, 48, and 144 for the experiments, and other parameters remained the same. The results are shown in Figure 9. From the results, we can observe that on all datasets, when the window size is increased, higher precision, recall, and $F1$-score can be obtained. This means that if the duration of the window is too short, the model may fail to learn that long-term anomalies have occurred. In QAR data, anomalies that occur during flight are more likely to be continuous segments than isolated points. This proves that our model structure can detect abnormal events for a longer period of time, and the data are relatively stable in the climb and cruise stages, which makes it more suitable for relatively large size windows to improve the detection efficiency.

4.2.2. Effect of Latent Variable z Dimension. In addition to the window length, we also investigate the link between $z_{\dim}$ for the dimension of latent variable z and detection performance. In VAE, the dimension of the latent variable space is a crucial parameter, which represents the important information required in the original data and can determine the representation ability of the latent space. VAE uses a probability distribution over the latent space to sample new data that can represent the characteristics of the original data. The embedding results obtained by sampling in different dimensions are different, and the reconstructed data are also very different. We set the dimensions of the latent variable z to 5, 10, 15, and 20 to observe its performance impact on the anomaly detection reconstruction process. Figure 10 shows the experimental results. The results show that if the latent variable z is located in a very large dimension, it will cause unnecessary redundancy to hinder the learning of the model, which may lead to the performance degradation of the VAE model training data. However, this does not mean that the smaller the latent variable space, the better. Considering that there is a special case, when the dimension is too small, VAE will lose a lot of information in the encoding stage and cannot decode. The model cannot fully capture the time dependency, resulting in poor model performance. It can be seen from the figure that the $F1$-score is relatively stable when the dimension is moderate. This confirms the above discussion. A suitable latent space size can make the model more robust in anomaly detection.

4.2.3. Effect of Head Number in MHSA Mechanism. In order to explore the effect of the number of heads on the model performance in the multihead self-attention mechanism, we set different head numbers of 2, 4, 6, and 8 for experiments. The experimental results are shown in Table 2. The results showed that in the climb and descent stages, the $F1$-score was the highest when head = 6. In the cruise phase, the $F1$-score is the highest when head = 4. Overall, the performance of the model fluctuates. As the number of heads increases, each head captures different aspects of information, and the model can capture more temporal information. The model performs the worst when there are only 2 heads, but an excessive number of heads makes the information captured between each self-attention head redundant, which weakens the model's ability to extract effective correlations. Combining the experimental results and efficiency, we set the number of heads to 6 in our implementation.

4.3. Analysis of Training Time. In this subsection, we also record the running time of epochs in each stage dataset and compare our method with several other deep learning hybrid models. All methods are compared on the same system. Table 3 shows the results obtained. The results show that our model is less time-consuming than other models, because we added a multihead self-attention mechanism to the LSTM. The parallel operation of multiple self-attention mechanisms can not only extract hidden features at a deeper level but also reduce the dimension and the amount of calculation. Therefore, we not only achieved good performance in anomaly detection but also reduced training time and improved operating efficiency.

4.4. The Comparisons of Using Different Datasets. In this subsection, to verify the feasibility of our method, we conduct experiments on several different public benchmark datasets. They are the KPI and NAB datasets that are often used to perform experiments in time series anomaly detection. Normal and abnormal are already marked in these datasets. The KPI dataset is from the AIOps Challenge held by Tsinghua University in 2018 [29]. Many Internet companies monitor the data generated by various performance indicators in order to ensure the stability of web services, such as CPU usage and server health, and other performance indicators. We randomly selected two time series from the KPI dataset for experiments. The NAB dataset, provided by artificial neural network company Numenta, contains a variety of streaming data in real-time applications, consisting of multiple labeled real-world and artificial time series data files. We selected the CPU usage of Amazon Web Services (AWS) servers and AWS EC2 servers collected by the Amazon Cloudwatch service as our dataset. Table 4 lists the data such as size, mean, standard deviation, and anomaly ratio of the four datasets, and it can be seen that these four datasets are significantly different.

We divided each data set into two parts: training set and test set, because our model needs to use normal data to train, so we removed the abnormality in the training data and got normal data. Outliers in the test set are reserved for testing.

Table 5 shows the experimental results. It can be clearly seen that our method outperforms other methods on these four public datasets. The accuracy of our model on these datasets is different. The $F1$-score of most datasets is above 0.9, and most datasets have achieved a 100% recall rate, which indicates that the number of false negatives (FN) is low. Because of the diversity of KPI and NAB datasets, some are cyclical, and some are unstable and fluctuating. This proves that our method performs well, can also detect different types of data anomalies, and has good generalization ability.

5. Conclusion

In this paper, we propose VAE-based MHSA-LSTM, an unsupervised deep learning-based method for anomaly detection in time series. The method can be divided into two stages. One is the model training stage. First, the variational autoencoder model is pretrained, and the features of normal data are learned, which can form stable local features in each window. The second is the anomaly detection stage, which uses the learning ability of the LSTM model for temporal representation and the feature extraction ability of the self-attention mechanism to identify anomalies based on the anomaly scores of the sample reconstruction calculation window. The VAE-based MHSA-LSTM combines encoder-decoder, generator, and multihead self-attention mechanism, which can detect all types of anomalies more comprehensively, quickly, and accurately. In the experimental part, we apply VAE-based MHSA-LSTM to the QAR dataset generated by real-world flights. Compared with several other classical reconstruction-based time series anomaly detection methods, the results show that our method has a better effect. In addition, we also applied our method on other public datasets with stable results.

Although our method achieves good performance and can accurately detect anomalies, there are still some limitations. Our model needs to be trained on the training data before anomaly detection, and the training set must ensure that there is no abnormal data. This presents some difficulties with the collection and processing of data. Therefore, in the future, we will explore the space for further development based on some of the ideas presented in this article.

Data Availability

The data used in this study are available from the corresponding author upon request.

Conflicts of Interest

The authors have no conflicts of interest to declare that are relevant to the content of this article.

Acknowledgments

This work was supported by the project of Natural Science Foundation of China (Nos. 61402329 and 61972456) and the Natural Science Foundation of Tianjin (Nos. 19JCYBJC15400 and 21YDTPJC00440).

References

[1] K. Mitchell, B. Sholy, and J. Alan, "General aviation aircraft flight operations quality assurance: overcoming the obstacles," *IEEE Aerospace and Electronic Systems Magazine*, vol. 22, no. 6, pp. 9–15, 2007.

[2] H. Heng, J. Zhang, and C. Xin, "Research on aircraft engine fault detection based on support vector machines," in *Proceedings of the International Conference on Consumer Electronics, Communications and Networks*, pp. 496–499, CECNet, Yichang, China, April 2012.

[3] T. Ergen and S. S. Kozat, "Unsupervised anomaly detection with LSTM neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 8, pp. 3127–3141, 2020.

[4] Bo Zong, S. Qi, R. Martin et al., "Deep autoencoding Gaussian mixture model for unsupervised anomaly detection," in *Proceedings of the International Conference on Learning Representations*, Vancouver, BC, Canada, May 2018.

[5] D. Park, Y. Hoshi, and C. C. Kemp, "A multimodal anomaly detector for robot-assisted feeding using an lstm-based variational autoencoder," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1544–1551, 2018.

[6] A. Vaswani, N. Shazeer, N. Parmar et al., "Attention is all you need," in *Proceedings of the Annual Conference on Neural Information Processing Systems*, pp. 5998–6008, Long Beach, California, USA, December 2017.

[7] M. Braei and S. Wagner, "Anomaly detection in univariate time-series: a survey on the state-of-the-art," *CoRR*, 2020, https://arxiv.org/abs/2004.00433.

[8] Q. Yu, L. Jibin, and L. Jiang, "An improved arima-based traffic anomaly detection algorithm for wireless sensor networks," *International Journal of Distributed Sensor Networks*, vol. 12, no. 1, 2016.

[9] T. B. Ottosen and P. Kumar, "Outlier detection and gap filling methodologies for low-cost air quality measurements," *Environmental Science-Processes & Impacts*, vol. 21, pp. 701–713, 2019.

[10] G. Münz, Sa Li, and G. Carle, "Traffic anomaly detection using k-means clustering," *GI/ITG Workshop MMBnet*, vol. 7, p. 9, 2007.

[11] L. Li, S. Das, R. John Hansman, R. Palacios, and A. N. Srivastava, "Analysis of flight data using clustering techniques for detecting abnormal operations," *Journal of Aerospace Information Systems*, vol. 12, no. 9, pp. 587–598, 2015.

[12] W. Zhao, L. Li, S. Alam, and Y Wang, "An incremental clustering method for anomaly detection in flight data," *CoRR*, vol. 132, Article ID 09874, 2020.

[13] C. Zeng, R. Wang, and Q. Zuo, "Analysis of abnormal flight and controllers data based on dbscan method," *Security and Communication Networks*, vol. 2022, Article ID 7474270, pp. 1–8, 2022.

[14] E. Smart, D. J. Brown, and J. Denman, "A two-phase method of detecting abnormalities in aircraft flight data and ranking their impact on individual flights," *IEEE Transactions on*

Intelligent Transportation Systems, vol. 13, no. 3, pp. 1253–1265, 2012.

[15] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[16] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, and T. Soderstrom, "Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding," in *Proceedings of the International Conference on Knowledge Discovery & Data Mining*, pp. 387–395, London, UK, July 2018.

[17] A. Khorram, M. Khalooei, and M. Rezghi, "End-to-end cnn+lstm deep learning approach for bearing fault diagnosis," *Applied Intelligence*, vol. 51, no. 2, pp. 736–751, 2021.

[18] A. Vahdat and J. Kautz, "NVAE: a deep hierarchical variational autoencoder," *Annual Conference on Neural Information Processing Systems (NeurIPS)*, vol. 33, pp. 19667–19679, 2020.

[19] D. Li, D. Chen, B. Jin, S. Lei, G. Jonathan, and N. G. See-Kiong, "MAD-GAN: multivariate anomaly detection for time series data with generative adversarial networks," in *Proceedings of the International Conference on Artificial Neural Networks*, pp. 703–716, Lausanne Switzerland, September 2019.

[20] K. Kishore, S. Sarkar, V. Venugopalan, and M. Giering, "Anomaly detection and fault disambiguation in large flight data: a multi-modal deep auto-encoder approach," *Annual Conference of the PHM Society*, vol. 8, 2016.

[21] M. Memarzadeh, B. Matthews, and I. Avrekh, "Unsupervised anomaly detection in flight data using convolutional variational auto-encoder," *Aerospace*, vol. 7, no. 8, 2020.

[22] Q. Wang, K. Qin, B. Lu, and R. Huang, "Feature extraction of qar data via sequence-parameter attention based convolutional autoencoder model," in *Proceedings of the IEEE 3rd International Conference on Civil Aviation Safety and Information Technology (ICCASIT)*, pp. 352–355, IEEE, Changsha, China, December 2021.

[23] O. I. Provotar, Y. M. Linder, and M. Maksym, "Unsupervised anomaly detection in time series using lstm-based autoencoders," in *Proceedings of the IEEE International Conference on Advanced Trends in Information Theory*, pp. 513–517, Kyiv, Ukraine, December 2019.

[24] D. Jimenez Rezende, S. Mohamed, and D. Wierstra, "Stochastic backpropagation and approximate inference in deep generative models," in *Proceedings of the International Conference on Machine Learning*, pp. 1278–1286, PMLR, Beijing, China, June 2014.

[25] Y. Wu, M. Schuster, Z. Chen et al., "Google's neural machine translation system: bridging the gap between human and machine translation," *CoRR*, 2016, https://arxiv.org/abs/1609.08144.

[26] F. T. Liu, K. M. Ting, and Z. H. Zhou, "Isolation forest," in *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2008)*, pp. 413–422, IEEE Computer Society, Washington, DC, USA, December 2008.

[27] Y. Wei, J. Jang-Jaccard, W. Xu, F. Sabrina, S. Camtepe, and M. Boulic, "Lstm-autoencoder based anomaly detection for indoor air quality time series data," *CoRR*, 2022, https://arxiv.org/abs/2204.06701.

[28] S. Lin, R. Clark, and R. Birke, "Anomaly detection for time series using vae-lstm hybrid model," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4322–4326, IEEE, Barcelona, Spain, https://ieeexplore.ieee.org/author/37297514400, Barcelona, Spain, May 2020.

[29] N. Zhao, J. Zhu, Y. Wang et al., "Automatic and generic periodicity adaptation for kpi anomaly detection," *IEEE Transactions on Network and Service Management*, vol. 16, no. 3, pp. 1170–1183, 2019.