Anomaly Detection of Multivariate Time Series Based on Metric Learning

Hongkai Wang[1], Jun Feng[2], Liangying Peng[1(✉)], Sichen Pan[1], Shuai Zhao[2], and Helin Jin[3]

[1] State Grid Zhejiang Information and Telecommunication Branch, Hangzhou, China
peng_liangying@zj.sgcc.com.cn
[2] Grid Zhejiang Information and Telecommunication Branch, Hangzhou, China
[3] Harbin Institute of Technology, Harbin, China

Abstract. Most of the current methods for anomaly detection in time series are unsupervised. However, unsupervised learning assumes the distribution of the data and cannot obtain satisfactory results in some scenarios. In this paper, we design a semisupervised time series anomaly detection algorithm based on metric learning. The algorithm model mines the features in the time series from the perspectives of the time domain and frequency domain. Furthermore, we design a loss function for anomaly detection. Different from the two-class loss function, in the scenario of the loss function we designed, the normal data will be clustered and distributed in the embedding space, and the abnormal data will be far from the normal data distribution. Furthermore, we extend our designed metric learning model to a semisupervised learning model, extending the labeled dataset with the unlabeled dataset by setting different confidence levels. We conduct experiments on different public datasets and compare them with commonly used time series anomaly detection algorithms. The results show that our model has a good effect. At the same time the semisupervised setting does improve the accuracy of model detection.

1 Introduction

In the scenario of anomaly detection of multivariate time series, many existing algorithms cannot achieve good results [8]. In particular, the distance-based anomaly detection algorithm, usually considers that a location with a low distribution density of points is more likely to be an anomaly. However, this kind of algorithm is very dependent on the definition of distance. The usual distance definition methods include Euclidean distance and Editing distance. However, this kind of distance definition method becomes inapplicable in the case of high-dimensional time series data. Euclidean distance does not consider the relationship between dimensions, and edit distance can only measure the distance of a single dimension, and then expand to multiple dimensions. These methods have poor performance in high-dimensional scenarios. Extending to high-dimensional

Y. Wang et al. (Eds.): ICPCSEE 2022, CCIS 1628, pp. 94–110, 2022.
https://doi.org/10.1007/978-981-19-5194-7_8

time series data, the situation becomes more complicated. In addition to individual data points, we also need to consider the temporary dependence that exists between data points. Therefore, anomaly detection in multidimensional time series is very challenging.

Today, popular anomaly detection models are usually unsupervised. In the time series of real scenes, label data are more difficult to obtain [2]. However, these unsupervised learning methods assume the distribution of the data and consider those points in the data where the distribution gathers as normal points. However, in practical scenarios, such a simple assumption can lead to many misjudgments. For example, there may also be some clustered outliers in the clustered distribution points. For example, in the data generated by wind turbines, anomalies include sparse outliers and stacked outliers. Stacked outliers are abnormal points in the aggregated distribution. Using traditional unsupervised anomaly detection methods will misjudge these data as normal points. However, at this time, we know the anomaly of this part of the point, and the unsupervised learning method cannot use this knowledge to help detect the anomaly.

Supervised learning can solve problems where data labels cannot be exploited. In real scenarios, prior knowledge can also be converted into labeled data to utilize this knowledge. The supervised learning anomaly detection algorithm has been researched in network intrusion detection, but less research has been conducted in other fields. On the one hand, data labels are difficult to obtain in real-world scenarios, and the cost of labeling datasets is too high; on the other hand, even if we have labeled datasets, it is difficult for us to obtain all types of anomalies and abnormal randomness. The label data required for supervised learning anomaly detection algorithms can be very large. Therefore, it is unrealistic to use supervised learning algorithms to solve the problem of anomaly detection in real scenarios [3]. Therefore, in summary, the challenges faced by anomaly detection in high-dimensional time series data are as follows:

- Currently, the dimension of time series is high and the number is large, and it is difficult for traditional anomaly detection methods to obtain good detection results.
- The unsupervised anomaly detection algorithm has a strong assumption about the data distribution, and considers that the points of the cluster distribution are normal points. However, the actual situation may be more complicated than this, and some abnormal points are not discretely distributed.
- Supervised learning anomaly detection algorithms can utilize knowledge in different fields, but the reality is that we cannot obtain such a large and complete data label, so supervised learning anomaly detection algorithms are not practical.

Based on these challenges, we propose a time series anomaly detection model based on metric learning, and propose a new loss function adapted to deep metric learning anomaly detection. Combining the advantages of CNN and LSTM, our model is a model that can effectively extract features from high-dimensional time series data and embed them into low-dimensional space. The model uses

random dimension permutation and short-time Fourier for feature extraction of raw time series. Among them, random dimension permutation can effectively mine feature associations between different dimensions, and add randomness to improve the robustness of the model. Short-time Fourier techniques can mine sequence-related information in the frequency domain. It also includes raw time series input into the LSTM module to mine short-term dependencies in time series data to form the final model. To better allocate the influence of the vectors of the embedding space of different dimensions on the center, we add an attention module to optimize the solution method of the category center in the place where the model finds the normal category center. Moreover, we propose a metric learning loss function suitable for anomaly detection, and experimentally verify that it is superior to loss functions such as cross entropy.

Deep metric learning is generally used to solve classification tasks [11]. There are many metric learning methods for classification tasks, but the basic idea is to make the distance between the data of the same category in the metric space as close as possible, and the distance between the data of different categories as far as possible. Such as the use of the prototype network [10]. However, this is not suitable for anomaly detection scenarios. Usually, anomaly detection can be regarded as a binary classification problem. However, there may be many types and causes of anomalies, so it is unreasonable to directly regard anomalies as one category. A point of view we agree with is that anomaly detection is a one-class classification problem [6], that is, judging whether the data are of a normal category. If the data are not of a normal class then it is turned into an anomaly. One-class SVM is an algorithm that thinks like this. Therefore, in the process of anomaly detection by metric learning, we only need to save a "center" of a normal category. The normal samples are as close as possible to this center, and the anomalies are as close as possible to this center [1]. The loss function we designed is based on this idea.

In addition, our model is also able to utilize a large amount of unlabeled data, extending to a semisupervised learning model. We can use a small amount of data to first learn the parameters of the model and the center of the class, and then use the unlabeled data to augment our dataset. At the same time, users can select different thresholds according to their needs to obtain data with different confidence levels. The final model only needs to use a small amount of data to achieve good results. In summary, the contributions of our paper can be summarized as follows:

- We propose a model that can extract the features of high-dimensional time series, and jointly mine data information from three perspectives: high-dimensional, time-series, and frequency domains.
- Based on the proposed network model, we propose a metric learning loss function suitable for anomaly detection. It can provide a low-dimensional distance space representation for time series. In this distance space, normal categories will be clustered together, and abnormal data will be far away from normal data clusters.

- We extend the model to semisupervised learning. The model can get good performance using only a small amount of labeled data and some unlabeled data.
- We conduct experiments on different public datasets and also compare supervised and semisupervised learning. The results show that both our model and the semisupervised setting have good performance.

The content of the following article is summarized as follows. First, a basic introduction and definition of some definitions, such as time series, anomaly scoring, etc. Then we mainly introduce our high-dimensional time series feature extraction model, which includes and processing modules, including segmentation, short-time Fourier transform on segments and random dimension permutation. Then the convolutional neural network, the recurrent neural network and the attention mechanism added to the center are assembled to form the final network model. Then we introduce the loss function of the model and the form of anomaly scoring, and extend the model to the semisupervised learning module. Finally, experiments are carried out on public datasets to verify the effectiveness and accuracy of the model.

2 Preliminaries

Time Series

A time series $Q = <\mathbf{q_1}, \mathbf{q_2}, \ldots, \mathbf{q_n}>$ is a chronologically organized sequence of vectors, each of which $\mathbf{q_i} = (s_i^{(1)}, s_i^{(2)}, \ldots, s_i^{(k)})$ represents the data generated at time t_i, where $1 \leq i \leq n$. And the vector sequence is arranged in chronological order, that is, when $i < j$, we have $t_i < t_j$. When $k = 1$, the time series is *univarate*, and when $k > 1$, the time series is *multivariate*. Here we mainly study the challenges brought by the current high-dimensional time series data, that is, the scenario of $k > 1$.

Anomalies and Anomaly Scoring in Time Series

Given a time series $Q = <\mathbf{q_1}, \mathbf{q_2}, \ldots, \mathbf{q_n}>$, our goal is to find those data points that are incorrect. To achieve this, for each point $\mathbf{q_i}$ in the sequence, we can calculate the anomaly score $OS(\mathbf{q_j})$ of it. The higher $OS(\mathbf{q_j})$ is, the more likely $\mathbf{q_i}$ is an anomaly.

Semisupervised Time Series Anomaly Detection

The current mainstream anomaly detection algorithms are unsupervised learning, and there are also a small number of supervised learning anomaly detection examples. However, semisupervised learning anomaly detection [9] is more suitable for use in real scenarios. In the context of semisupervised learning for multidimensional time series anomaly detection, we believe that the labels used

in training the model are not sufficient. That is, only a portion of the data has labels. Therefore, we plan to use our model to label unlabeled data so that these data can also be used in supervised learning multidimensional time series anomaly detection algorithm. Therefore, the dataset used in the training process is a set of tuples, $((\mathcal{Q}, \boldsymbol{y}), \widetilde{\mathcal{Q}}))$, where $\mathcal{Q}$ has the label set $\boldsymbol{y}$, while $\widetilde{\mathcal{Q}}$ has no label. In order to solve the problem of insufficient datasets, the usual semisupervised learning multidimensional sequence anomaly detection problem first uses the labeled dataset $(\mathcal{Q}, \boldsymbol{y})$ to train a classifier $\tilde{f}_Q \mapsto y$, and then use this classifier to label our unlabeled dataset $\widetilde{\mathcal{Q}}$, and finally get an expanded dataset. The problem can then be treated as a supervised learning task.

3 Proposed Model

This section introduces our model. The first is data preprocessing, including segmentation, short-time Fourier transform and random dimension permutation. The preprocessing part allows the model to better extract features from the data. Then there is our proposed network model, which combines different inputs through convolution and LSTM and other structures to extract features at different levels, and finally combines them to form the final embedding vector. Then comes the loss function part of the model, in which we propose a loss function based on metric learning for anomaly detection. Then combined with the loss function, it illustrates how the model scores anomalies. Finally, the model is extended to the form of semisupervised learning, making it effective in detecting anomalies even with a small number of samples. The overall flow of the model is shown in Fig. 1.

3.1 Preprocessing

The first step is to segment the data. Using average segmentation similar to PAA [4] will further reduce our limited data, and at the same time, it is not good enough for continuous features of time series data. Therefore, to better obtain data from small labeled samples, and at the same time to allow the segmented data to retain the continuity in the data as much as possible, we consider direct coverage between segments during segmentation. If the specified segment size is s, we define a coverage rate of τ, and unlike the average segment, we let two adjacent segments have $\lceil \tau \times s \rceil$ is identical. For example, the sequence $<1, 2, 3, 4, 5, 6, 7, 8>$. If the segment size is 4, then the result of the segment is $<\{1, 2, 3, 4\}, \{3, 4, 5, 6\}, \{5, 6, 7, 8\}>$. After segmenting, we obtain a matrix of $n \times s \times d$, where n is the number of segments, s is the segment length, and d is the dimension of the data point.

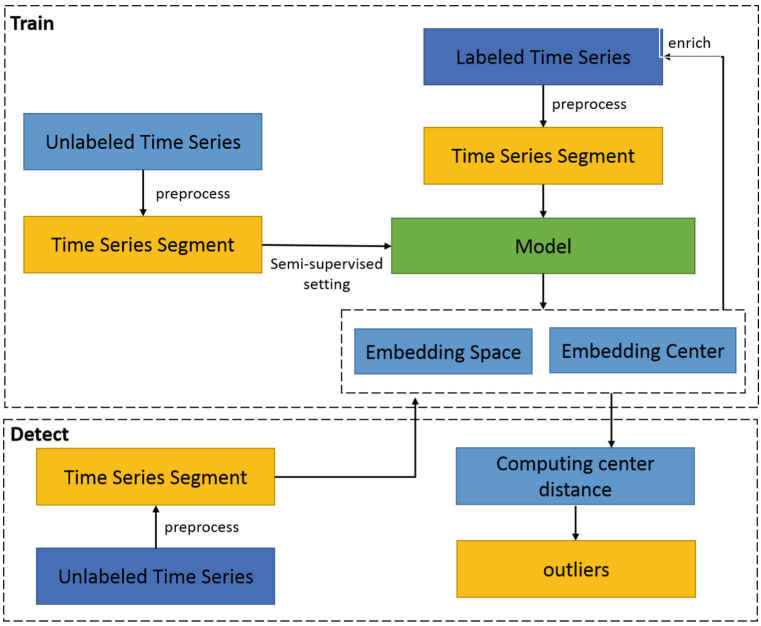

Fig. 1. Overall flow

We preprocess the obtained sequence of data segments, we preprocess it to extract features from the data. The ones used here include random dimension permutation and short-time Fourier transform. Random dimension permutation is a method that can mine the correlation between different dimensions of multi-dimensional time series. The short-time Fourier transform can obtain the frequency domain signal corresponding to the data segment, which can provide a new perspective for anomaly detection in many cases. The specific method is as follows:

Random dimension permutation can mine the associations between different dimensions in multi-dimensional time series, and can also effectively mine the patterns contained in some dimensions. We extend it again so that it can accommodate time-series data segments. Assuming that our time series dimension is k, we want to divide each data segment into g groups. Then we can calculate the dimension size of the data in each group as $\varphi = \left\lfloor \frac{m \cdot \alpha}{g} \right\rfloor$, where φ is a parameter that controls the size of the group, and $\lfloor \cdot \rfloor$ is the symbol for rounding down. Each random dimension permutation process needs to randomly arrange the dimensions of the data, and then select the first φ dimensions as the result obtained this time. In the data segment scenario, we only need to randomly arrange each dimension of the data segment according to the same arrangement rule, and finally for each data segment, we can obtain a $g \times s \times \varphi$ size matrix. An example of random dimension permutation is shown in Fig. 2.

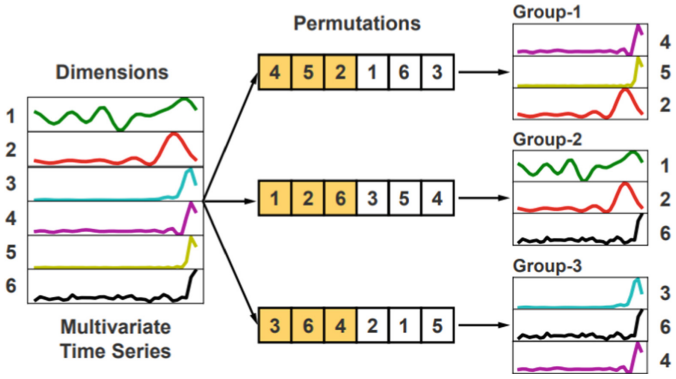

Fig. 2. An example of random dimension permutation

When we deal with time series, the data we obtain may come from different locations and types of sensors, which makes us obtain data that are not synchronized. However, the resulting pattern will be embodied in a certain "shape", when we transform the data from the time domain to the frequency domain, which will give the model a new perspective to mine the data. Here we use the short-time Fourier transform [5]. The research of Li et al. shows that the short-time Fourier transform can effectively extract the features of time series data in the time domain, and then substitute it into the neural network as the preprocessed data for the next step of training. For the time series segment obtained by the previous processing, the mth data in the nth data can be converted into:

$$STFT^{(\tau,s)}\{\mathbf{x}\}(m,n) = \sum_{t=1}^{T} \mathbf{x}(t) \cdot \mathbf{w}(t - s \cdot m) \cdot e^{-j\frac{2\pi n}{\tau}(t - s \cdot m)}$$

where $STFT^{(\tau,s)}$ represents our short-time Fourier transform function with parameters τ and s. The difference between the Fourier transform and the short-time Fourier transform is that the short-time Fourier transform performs the Fourier transform on the local segments after the data are segmented. In our data preprocessing process, the data has been segmented, so only the Fourier transform on the segment is equivalent to the short-time Fourier transform.

The two preprocessing processes are from different aspects, one is to extract features between different dimensions in high-dimensional data, and the other is to mine more information in the frequency domain in single-dimensional time series data, and consider substitute the original data into the model, and finally form all the inputs of the model.

3.2 Encoder for High-Dimensional Time Series Data

The main purpose of this part of the encoder is to make a low-dimensional representation of a high-dimensional time series data segment for further processing and use. TapNet is an effective high-dimensional time series feature extraction method [12], unlike theirs, our model is calculated in segments, which is very good for data scale expansion capability. Moreover, the short-time Fourier transform module is added to our module, which can extract more information from the frequency domain.

The network model is mainly divided into three parts, one of which is to flatten the raw time series segments, and then the module for feature extraction through LSTM. One part is to perform a short-time Fourier transform on time series data and then perform a one-dimensional convolution to extract features. The last part is a process of randomly extracting time series data, that is, random dimension permutation, and then performing one-dimensional convolution in different dimensions. Among them, the LSTM module is proposed to better extract time series features in the time series. The purpose of flattening the original time series segments is to make the data fit the input of the LSTM module. In this part, our input is $X \in \mathcal{R}^{s \times d}$, and the size of the output is the size of the latent space of the LSTM module, i.e. $X_{lstm} \in \mathcal{R}^{s \times h_{lstm}}$.

Then, methods based on short-time Fourier transform and random dimension extraction are both based on convolutional neural networks. Because of the particularity of time series, the convolutional neural networks we use are all one-dimensional. Moreover, batch norm and ReLU activation functions are added after each convolutional layer to better extract features, and finally a maxpool module is used for pooling. The short-time Fourier module changes the thinking and extracts the features of the time series in the frequency domain. Research shows that such feature extraction is effective. Then, after performing the short-time Fourier transform module on the time series segment, the convolution module can also obtain an output $X_{SFTF} \in \mathcal{R}^{s' \times h_{sftf_{conv}}}$.

The random dimension permutation module is designed to randomly extract some dimensions in the data multiple times, which can not only randomly learn the information interaction between different dimensions, but also can improve the robustness of the model. The specific operation process is similar to TapNet. For each decimation, we obtain an output $X^{(i)_{random_{conv}}} \in \mathcal{R}^{d_f \times 1}$. Finally, the results of these three parts are expanded and spliced to obtain a synthetic long vector. Then this long vector is merged and dimensionally reduced through a fully connected layer to obtain the final embedding space vector. Then it can be trained or predicted according to different scenarios (Fig. 3).

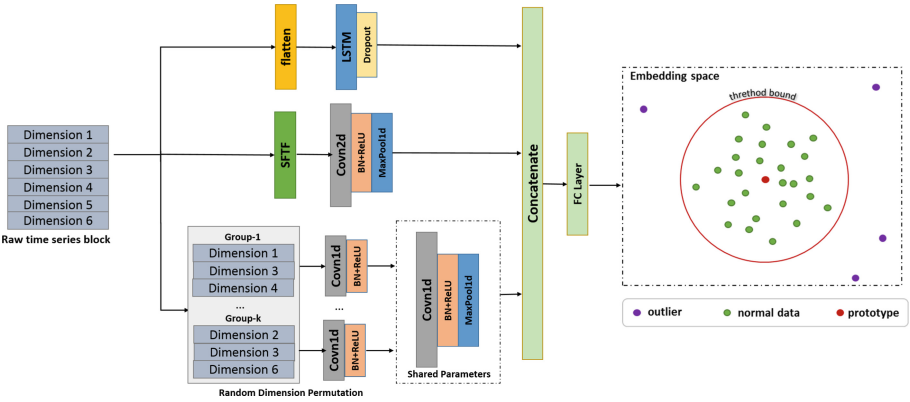

Fig. 3. Model

3.3 Attentional Center Learning

The main goal of our learning is the parameters of the network model, and the center of the normal data in the embedding space. Because we can only work with limited data, there may be an inductive bias in the learning of the network model [7]. Therefore, we added an attention module to the last center learning, that is, attentional center learning. This attention mainly solves that the normal data and abnormal data may focus on different embedding vectors when calculating, that is, the contribution of different dimensions to the center calculation in the embedding space may be different, and each dimension may not be the same. Want to wait for the weight.

The idea of the attention module is as follows: when calculating the center $\mathbf{c}$, attention can be added to each dimension, which is formulated as $\mathbf{c} = \sum_i A_i \cdot Hi$, where A is the attention vector of the embedding dimension of the center, and H is all datasets of the normal category. A is the parameter to be learned through learning, $A = softmax(\mathbf{w}^T tanh(VH^T))$, where w and V are both learned during the learning process model parameters. If this attention module is added, after the embedding vector is finally obtained, an attention operation must be performed on the vector to obtain the final embedding vector.

3.4 Loss Function

Intuitively, the anomaly detection problem can be regarded as a binary classification problem, but it is somewhat different from the traditional binary classification problem. However, there are only two types of result labels for anomaly detection, due to the randomness and diversity of outliers. Therefore, we do not solve it as a binary classification problem, but an "either-or" single-class detection. We only need to learn the category features of normal data as much as possible. If the data have a large deviation from the normal category, we can treat it as an anomaly. Therefore, our loss function is formulated as follows:

$$loss = \frac{1}{|Q_p| + |Q_o|}[\sum_{\mathbf{x} \in Q_p} d(f_\phi(\mathbf{x}), \mathbf{c}_s) + \alpha \sum_{\mathbf{x} \in Q_o} (md(f_\phi(\mathbf{x}), \mathbf{c}_s))]$$

where $\mathbf{c}_s = \frac{1}{|S_p|} \sum_{(\mathbf{x},y) \in S_p} f_\phi(\mathbf{x})$, S_p is a sample; Q_s is a query sample dataset of normal data, Q_o is a query sample dataset of abnormal data, $f_\phi : \mathcal{R}^D \to \mathcal{R}^M$ is the embedding function learned by the neural network, ϕ is the parameter set to be learned by the neural network; α is the parameter used to balance the two distance weights, $d : \mathcal{R}^M \times \mathcal{R}^M \to [0, +\infty)$ is a distance method to measure the similarity of two vectors. This is the Euclidean distance, because the specific features are learned by the network model, so the form of the definition is not important. The meaning of the final loss function definition is to make the normal data as small as possible from the class center, and make the abnormal data as large as possible from the class center. The parameters finally learned by the model are the parameter set ϕ learned by the neural network and the category center $\mathbf{c}_s$ of the normal data.

In addition, the parameters of the model and the center of the embedding space are updated. Because the data brought in by our method for each training are the data segment at this moment, the data segment at the next moment cannot be brought into the center of the data segment completely as the center of the whole, which is easily destroyed due to anomalies. The stability of the model. For the center $\mathbf{c}_{t-1}$ obtained at the previous moment, and the center $\mathbf{c}'_t$ obtained from the normal data after the training of the current data segment, the calculation method of the final center is: $\mathbf{c}_t = \mathbf{c}_{t-1} + \beta \mathbf{c}'_t$, that is, the calculation of the center is incremental. Therefore, our model can also be easily extended to the mode of online algorithms. This is also one of the strengths of our model. The model training process can be shown as Algorithm 1.

Algorithm 1 . The process of training the model, where $SUBSERIES(Q, start, end)$ represents the subsequence of the sequence Q from $start$ to end

Require: Training dataset $Q =< (\mathbf{q_1}, y_1), (\mathbf{q_2}, y_2), \ldots, (\mathbf{q_n}, y_n) >$, where $y_i \in 0, 1$, 0 means normal data, 1 means abnormal data, window size s, coverage rate τ, class center obtained in the previous round $\rfloor_{t-1}$

Ensure: Backpropagate the result of the loss trained on the current t batch and obtain the new class center c_t

$Q_t \leftarrow SUBSERIES(Q, s + (1-\tau)s(t-1), 2s + (1-\tau)s(t-1))$
$J \leftarrow 0$
for $(\mathbf{x}, y) \in Q_t$ **do**
  $J = J + (1-y)d(f_\phi(\mathbf{x}), \mathbf{c}_{t-1}) + \alpha y(m - d(f_\phi(\mathbf{x}), \mathbf{c}_{t-1}))$
end for
$J \leftarrow \frac{J}{|Q_t|}$
Backpropagation J
for $(\mathbf{x}, y) \in Q_t and y = 0$ **do**
  $\mathbf{c}_t = \mathbf{c}_{t-1} + \beta f_\phi(x)$
end for

During the detection process, we judge the degree of abnormality of the data by scoring the abnormality of the data. The higher the abnormality score of a data segment is, the higher the probability that the data are abnormal. After training the neural network, we obtain the embedding function f_ϕ we need and the class center $\mathbf{c}_s$ of the normal data. The logic of the model we finally obtain is as follows: if the data segment is relatively close to the category center $\mathbf{c}_s$ after passing through the encoder, then the possibility of being abnormal is relatively small, otherwise it means that the possibility of being abnormal is relatively larger. Based on this, the final use of f_ϕ to calculate the abnormal score is mixed with the distance measurement method used in the previous training $d : \mathcal{R}^M \times \mathcal{R}^M \to [0, +\infty)$, if the defined method is Euclidean distance $d(x, y) = ||xy||_2^2$, then the abnormal score of data segment x is $OS(x) = ||f_\phi(x) - c_t||_2^2$.

3.5 Semisupervised Learning

If the number of labels provided is extremely low, our algorithm may repeatedly resample on small datasets, which may lead to overfitting problems. Some methods to solve the problem of overfitting due to few samples, such as data augmentation and adding noise to the data, have good results in the field of image processing. However, the time series data we deal with are often relatively simple real-valued data. Although the dimension is high, the data between different dimensions have their own unique meaning. These methods are often not applicable to these data. However, considering that our data often come from sensor networks, sensor networks will bring us much unlabeled data. We can generate some pseudolabels on these data to enrich our training dataset to prevent overfitting. Purpose. At this time, the small sample problem is transferred from the original only small sample data to a method similar to semisupervised learning. Let us introduce our simple semisupervised learning method using self-training based on the model learned from small samples.

For unlabeled datasets, the set U is obtained using the previous data preprocessing method, and then we calculate a confidence value for each data through the embedding function f_ϕ that has been learned earlier. Because the size of the anomaly score can reflect the probability that the test data are anomalous, we can simply use $confidnce(x) = \frac{1}{OS(x)}$ as the confidence value of the data segment. Then define a ratio γ, and add these data segments with high confidence as normal data to the training dataset to continue training. The specific algorithm for expanding the training dataset is shown in Algorithm 2.

Here, we consider that in the semisupervised setting in the anomaly detection scenario, most of the labeled samples are normal samples, and because there are more normal samples, the correct rate of identifying normal samples is higher than that of abnormal samples, that is, the model is more sensitive to normal categories. The sample recall is high, and the recall for anomalous categories is low. Therefore, we have reason to believe that if we identify an unlabeled sample as an anomaly in a semisupervised setting, then it should be an anomaly with a high probability, because we have seen a lot of normal data during training, in a sense, the judgment of normal data is in a state of "overfitting". Therefore,

Algorithm 2. Augment the dataset with self-training

$getTopK(L, k)$ is to get the first k elements in the list L
Require: Unlabeled dataset U, confidence ratio γ
Ensure: The augmented dataset D
$Confidence \leftarrow \frac{1}{OS(U)}$
$Confidence \leftarrow sort(Confidence, order = descending)$
$threshold \leftarrow min(getTopK(confidence, \gamma|U|))$
$D \leftarrow D \cup \{x \in U | \frac{1}{OS(X)} < threshod\}$

the data obtained in such a scenario is still abnormal with a high probability of being abnormal, so we can give it a higher weight in training. Because our model is judged by anomaly scoring, and there is no clear boundary between categories, this idea can be implemented in another way: set two thresholds, one is the non-abnormal threshold, and the normalized range is $0-\gamma_{normal}$, the other is the abnormal threshold, the range after normalization is $\gamma_{outlier}-1$, where the value of γ_{normal} can be set to be harsher (as small as possible), because we have enough normal data, so setting the threshold can ensure that the model learns the correct normal data; $\gamma_{outlier}$ can be set looser, such as 0.6 and 0.7, because we have learned enough The normal data can still obtain a large abnormal score under the condition of such category imbalance, indicating that it is very likely to be abnormal data. The overall improved process is shown in Algorithm 3.

Algorithm 3. Augment the dataset with improved self-training

$getTopK(L, k)$ is to get the first k elements in the list L, $getLastK(L, k)$ is to get the last k elements in the list L
Require: Unlabeled dataset U, confidence ratios γ_{normal} and $\gamma_{outlier}$
Ensure: The augmented dataset D
$Confidence \leftarrow \frac{1}{OS(U)}$
$Confidence \leftarrow sort(Confidence, order = ascending)$
$threshod_{normal} \leftarrow min(getTopK(confidence, \gamma_{normal}|U|))$
$threshod_{outlier} \leftarrow min(getLastK(confidence, \gamma_{outlier}|U|))$
$D \leftarrow D \cup \{x \in U | \frac{1}{OS(X)} < threshod_{normal}\} \cup \{x \in U | \frac{1}{OS(X)} > threshod_{outlier}\}$

4 Experiments

4.1 Dataset

The datasets used here are the oil chromatography datasets provided by the State Grid and some public datasets. There is no abnormality in the oil chromatography data, only different data states, so here we consider inserting some abnormal intervals into the data. These abnormal intervals increase or decrease their values on the basis of the original data, and then mark them as abnormal.

There are abnormal and normal data labels in public datasets. Here, some high-dimensional and single-dimensional time series anomaly detection datasets are mainly selected to suit the scenarios of our method. Included here are the server machine dataset and the ECG Dataset. Among them, server machine dataset is a high-dimensional time series dataset, which is the data collected by the author in multiple scenarios. The ECG dataset is a dataset in the field of electrocardiography, most of which are single-dimensional datasets, which are also in the field of time series anomaly detection. Commonly used datasets.

4.2 Setup

Evaluation Metrics

A measure of the accuracy of the results was obtained by calculating the ROC-AUC and PR-AUC. In the anomaly detection scenario, because the anomalies in the data are usually few, the results obtained by directly using the accuracy rate are meaningless. Using the AUC value can comprehensively consider the precision and recall rate, which is more practical. This measurement method comprehensively considers TP, FP, TN, FN in the binary classification problem, and sets various thresholds.

Hyperparameter Settings

The settings of hyperparameters are mainly distributed on the related settings of the neural network and the related settings of preprocessing. First, in our experimental setting, the coverage rate $\tau = 0.5$, the random dimension permutation is set to three groups, and each group randomly selects half of the currently used dataset dimensions. The window size is set to 20. In the settings related to the neural network, the vector dimension of the finally obtained embedding space is 64.

4.3 Result

Accuracy

First, the experimental results on the dataset provided by the grid are presented. Because there are no anomalies in the dataset, we insert a certain percentage of anomalies into the dataset to conduct experiments. Detect one or more of these segments as anomalies by numerically increasing them. The first experiment is the effect of different modules on the experimental results. We remove different modules to test the accuracy. The results of ROC-AUC are shown in Table 1, and the results of PR-AUC are shown in Table 2.

Table 1. ROC-AUC results on the grid dataset

Abnormal increase rate	ROC-AUC			
	Our model	No preprocessing	Use CNN only	Use LSTM only
12%	1.00	1.00	1.00	1.00
10%	1.00	0.972	0.987	0.982
7%	0.999	0.953	0.954	0.962
5%	0.995	0.948	0.940	0.953
2%	0.550	0.562	0.463	0.623

Table 2. PR-AUC results on the grid dataset

Abnormal increase rate	PR-AUC			
	Our model	No preprocessing	Use CNN only	Use LSTM only
12%	0.999	0.999	0.999	0.999
10%	0.999	0.955	0.967	0.962
7%	0.996	0.932	0.962	0.982
5%	0.958	0.962	0.951	0.942
2%	0.149	0.253	0.213	0.153

In addition, we also verified the effect of our loss function, which is mainly compared with cross entropy. As a classic binary classification loss function, cross entropy is well represented. The results are shown in Table 3.

Table 3. Comparison of loss functions on power grid data

Abnormal increase rate	ROC-AUC		PR-AUC	
	Our model	Cross entropy	Our model	Cross entropy
12%	1.00	0.999	0.999	0.999
10%	1.00	0.983	0.999	0.965
7%	0.999	0.981	0.996	0.921
5%	0.995	0.965	0.958	0.915
2%	0.550	0.532	0.149	0.135

Then, we also validate the effectiveness of our method on public datasets. The datasets used are server machine Dataset and ECG Dataset. Similarly, we first verify the effectiveness of different modules on public datasets for different modules, and the results are shown in Table 4 and Table 5.

Table 4. ROC-AUC results on the public dataset

Dataset	Our model	No preprocessing	Use CNN only	Use LSTM only
SMD 1	0.862	0.832	0.852	0.823
SMD 2	0.936	0.895	0.871	0.891
ECG1	0.968	0.952	0.912	0.935
ECG2	0.996	0.992	0.992	0.996

Table 5. ROC-AUC results on the public dataset

Dataset	Our model	No preprocessing	Use CNN only	Use LSTM only
SMD 1	0.776	0.786	0.723	0.767
SMD 2	0.852	0.811	0.843	0.821
ECG1	0.891	0.812	0.863	0.827
ECG2	0.683	0.694	0.563	0.593

In addition, we compare our method with other methods on public datasets, including the deep learning time series anomaly detection model LSTM autoencoder, the classic anomaly detection method isolation forest and one-class SVM. The results are shown in Table 6 and Table 7.

Table 6. Comparative experiments on public datasets(ROC-AUC)

Dataset	Our model	Isolated forest	LSTM-AE	oc-SVM
SMD 1	0.862	0.847	0.842	0.755
SMD 2	0.936	0.863	0.925	0.879
ECG1	0.968	0.935	0.963	0.924
ECG2	0.996	0.912	0.872	0.885

Table 7. Comparative experiments on public datasets(PR-AUC)

Dataset	Our model	Isolated forest	LSTM-AE	oc-SVM
SMD 1	0.776	0.426	0.538	0.688
SMD 2	0.852	0.723	0.785	0.256
ECG1	0.891	0.912	0.852	0.798
ECG2	0.683	0.523	0.292	0.463

As mentioned earlier, semisupervised learning algorithms can effectively use unlabeled data in the data to improve the ability of the model. Here we mainly discuss whether the use of semisupervised learning algorithms has a positive impact on the results. Because labeled data are a small part of the data, we

use a 2:8 ratio to divide labeled data and unlabeled data, and then compare the metric learning using only labeled data with our semisupervised learning algorithm using unlabeled data, respectively. The results are shown in Table 8.

Table 8. Comparative experiments on public datasets(PR-AUC), where SL indicates supervised learning and UL indicates unsupervised learning

Abnormal increase rate	ROC-AUC		PR-AUC	
	SL	UL	SL	UL
12%	1.00	1.00	0.999	0.999
10%	0.992	1.00	0.999	0.999
7%	0.972	0.999	0.954	0.996
5%	0.952	0.995	0.921	0.958
2%	0.425	0.550	0.135	0.149

5 Conclusion

For the high-dimensional time series anomaly detection scenario, our paper designs a semisupervised time series anomaly detection algorithm based on metric learning. First, unique preprocessing is performed for high-dimensional time series. It includes feature extraction of time series data, correlation extraction before different dimensions of high-dimensional data, and conversion to frequency domain mining of time series data features. Then, different from binary classification, we design a new loss function suitable for anomaly detection. In the feature space trained by this loss function, normal samples will be clustered and distributed, and abnormal data will be scattered and distributed far away from normal samples. We also extended the model to semisupervised learning. Only a small number of labeled and unlabeled samples are required to obtain good results. We have conducted experiments on power grid datasets and public datasets, including the comparison of the functions of different modules within the model and the comparison of different methods. The results show that our method can achieve good anomaly detection results.

Future work may continue to use metric learning models, extending the models to online learning. Additionally, research on anomaly detection algorithms for small sample time series will be considered. The data augmentation method mentioned in the paper is a solution, but few-shot learning encounters more problems.

Acknowledgments. The project is supported by State Grid Research Project "Study on Intelligent Analysis Technology of Abnormal Power Data Quality based on Rule Mining" (5700-202119176A-0-0-00).

References

1. Booth, B.G., Sijbers, J., Keijsers, N.L.W.: Outlier detection for foot complaint diagnosis: modeling confounding factors using metric learning. IEEE Intell. Syst. **36**(3), 41–49 (2021)
2. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: a survey. ACM Comput. Surv. (CSUR) **41**(3), 15 (2009)
3. Ezeme, O.M., Mahmoud, Q.H., Azim, A.: A framework for anomaly detection in time-driven and event-driven processes using kernel traces. IEEE Trans. Knowl. Data Eng. **34**(1), 1–14 (2022)
4. Fotso, V.S.S., Nguifo, E.M., Vaslin, P.: Grasp heuristic for time series compression with piecewise aggregate approximation. RAIRO Oper. Res. **53**(1), 243–259 (2019)
5. Li, S., Hong, D., Wang, H.: Relation inference among sensor time series in smart buildings with metric learning. In: The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, 7–12 February 2020, pp. 4683–4690. AAAI Press (2020)
6. Manevitz, L.M., Yousef, M.: One-class SVMs for document classification. J. Mach. Learn. Res. **2**(Dec), 139–154 (2001)
7. Neyshabur, B., Tomioka, R., Srebro, N.: In search of the real inductive bias: on the role of implicit regularization in deep learning. In: Bengio, Y., LeCun, Y., (eds.) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, 7–9 May 2015. Workshop Track Proceedings (2015)
8. Riffo, V., Mery, D.: Automated detection of threat objects using adapted implicit shape model. IEEE Trans. Syst. Man Cybern.: Syst. **46**(4), 472–482 (2015)
9. Ruff, L., et al.: Deep semi-supervised anomaly detection. In: ICML Workshop on Uncertainty Robustness in Deep Learning (2019)
10. Snell, J., Swersky, K., Zemel, R.S.: Prototypical networks for few-shot learning. In: Guyon, I., et al. (eds.) Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, Long Beach, CA, USA, 4–9 December 2017, pp. 4077–4087 (2017)
11. Sun, P., Yang, L.: Low-rank supervised and semi-supervised multi-metric learning for classification. Knowl. Based Syst. **236**, 107787 (2022)
12. Zhang, X., Gao, Y., Lin, J., Lu, C.T.: TapNet: multivariate time series classification with attentional prototypical network. In: The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, 7–12 February 2020, pp. 6845–6852. AAAI Press (2020)