frontiers | Frontiers in Physics

TYPE Original Research
PUBLISHED 06 January 2026
DOI 10.3389/fphy.2025.1704495

OPEN ACCESS

EDITED BY
Yuanyuan Huang,
Chengdu University of Information Technology, China

REVIEWED BY
Amin Ul Haq,
University of Electronic Science and Technology of China, China
Ashwini Rao,
SVKM's Narsee Monjee Institute of Management Studies, India

*CORRESPONDENCE
Tenglong Xie,
dsqyy124@163.com

RECEIVED 13 September 2025
REVISED 14 November 2025
ACCEPTED 18 November 2025
PUBLISHED 06 January 2026

CITATION
Xie T, Li B, Zhen Y, Wei H, Xu K and Huang J (2026) Anomaly detection method for power dispatch streaming data based on adaptive isolation forest and self-supervised learning. *Front. Phys.* 13:1704495.
doi: 10.3389/fphy.2025.1704495

Anomaly detection method for power dispatch streaming data based on adaptive isolation forest and self-supervised learning

Tenglong Xie*, Bo Li, Yingkai Zhen, Hao Wei, Kunhuan Xu and Jingyin Huang

Information Center of Guangdong Power Grid Co., Ltd., Guangzhou, China

Introduction: To address the issues of concept drift and scarcity of anomaly samples in real-time anomaly detection under the massive streaming data environment of power dispatching and control systems, this study focuses on developing an effective detection method.
Methods: We propose a streaming data anomaly detection method integrating adaptive isolation forests and self-supervised learning. First, a business-based model is constructed by analyzing inherent relationships between system services, processes, and resource usage. An improved isolation forest algorithm with a sub-forest progressive update mechanism is designed—selectively eliminating sub-detectors with large anomaly rate deviations and dynamically adding new ones to overcome performance degradation from traditional random updates. Additionally, a GPT-based self-supervised learning framework is introduced, incorporating state memory units to encode historical data patterns and a distance metric-based sampling strategy to reduce redundancy.
Results: Experiments on a real power dispatching process resource dataset show the proposed method significantly outperforms the traditional streaming data isolation forest algorithm in key indicators such as AUC value, with the highest improvement reaching 39.12%. Ablation experiments verify the effectiveness of each module.
Discussion: The proposed method enhances the detection algorithm's adaptability to concept drift, overall stability, and ability to perceive hidden anomalies, providing reliable technical support for the safe and stable operation of the power dispatching system.

KEYWORDS

adaptive isolation forest, self-supervised learning, power dispatch system, streaming data, anomaly detection

1 Introduction

As the nerve center of the smart grid, the stable and reliable operation of power dispatching and control systems is directly related to the safety and economic efficiency of the entire power system. With the continued expansion of

the power grid, the widespread integration of distributed energy resources, and the deepening of demand-side management, the complexity of power dispatching and control systems is increasing, and the scale and number of functional modules are growing exponentially. In this context, system failures caused by software process anomalies, resource contention, and data jumps are becoming increasingly frequent. These failures are often hidden in massive amounts of real-time operational data, making them difficult to detect and locate in a timely manner using traditional monitoring methods.

Especially in the streaming data environment generated by this system, data has core characteristics such as continuous high-speed arrival, possible concept drift, and extremely scarce anomalous samples. Traditional anomaly detection methods based on static thresholds set by expert experience are not only highly subjective and poorly adaptable, but also difficult to capture the collaborative and competitive relationships between complex processes and cannot effectively respond to dynamic changes in data distribution [12]. Therefore, the research of an intelligent anomaly detection algorithm that can adaptively update and fully exploit the internal temporal patterns and semantic features of streaming data has crucial theoretical significance and engineering value for achieving early warning of power dispatch control system failures and improving the reliability of power grid operations.

This paper addresses these challenges and aims to investigate novel anomaly detection methods for the streaming data environment of power dispatching and control systems. First, we analyze the inherent connections between business, processes, and resource usage within the system, laying the foundation for anomaly detection. Furthermore, the core contributions of this paper lie in two aspects: First, we propose an adaptive isolation forest-based anomaly detection algorithm for streaming data. This algorithm dynamically maintains a collection of detectors through a novel sub-forest progressive update mechanism, effectively overcoming the model performance degradation caused by random updates in traditional methods, while balancing detection efficiency and adaptability to concept drift. Second, to further mine the rich temporal information contained in streaming data, we introduce a self-supervised learning framework based on the GPT architecture, constructing state memory units to encode historical data patterns, thereby enhancing the model's understanding of normal behavior patterns and improving its ability to identify hidden anomalies.

Experimental validation on a set of real-world power dispatch process resource data demonstrates that the proposed method significantly outperforms traditional baseline algorithms in terms of overall performance (AUC value). Ablation experiments also demonstrate the effectiveness of each innovative module. This research provides new technical ideas and solutions for the safe and stable operation of power dispatch systems.

2 Related work

Based on their core concepts, anomaly detection technologies can be broadly categorized into statistical, distance, density, clustering, and learning-based approaches. Adapting these technologies to streaming data environments and meeting the high real-time and high reliability requirements of power dispatching and control systems remains a key focus and challenge in current research.

2.1 Traditional anomaly detection algorithms and their streaming adaptations

Early anomaly detection mostly relied on statistical methods (such as the 3σ principle and Grubbs test) or threshold methods based on expert experience. Although these methods are simple and efficient, they cannot capture the complex relationships between multidimensional features, and static thresholds are difficult to adapt to the dynamically changing environment of streaming data. To cope with the continuous arrival of streaming data, researchers have proposed many incremental learning algorithms. For example, the incremental version of the distance-based algorithm [1] (such as K-NN) requires dynamic maintenance of the nearest neighbor set, which has huge computational overhead; and the density-based algorithm [2] (such as LOF) also faces the high complexity problem of recalculating neighbor relationships during incremental updates. These methods are difficult to meet the stringent requirements for detection efficiency in power dispatch scenarios.

2.2 Isolation forest algorithm and its application to streaming data

The Isolation Forest (iForest) algorithm has significant advantages in processing high-dimensional big data due to its linear time complexity and the fact that it does not require distance calculation. It isolates samples through a random segmentation strategy, and outliers have a shorter path length due to their "easy to isolate" characteristics. The classic Isolation Forest algorithm is designed for offline batch data, and its direct application to streaming data will lead to performance degradation due to concept drift. To this end, literature [3] proposed a streaming data Isolation Forest algorithm that achieves incremental updates of the model by randomly replacing isolation trees. However, this random update strategy lacks specificity, which may unintentionally discard subtrees that perform well, leading to performance fluctuations and a decrease in overall detection accuracy. In contrast, the adaptive update mechanism proposed in this paper addresses this issue by selectively eliminating sub-detectors with the largest anomaly rate deviation. This targeted approach ensures that only poorly performing sub-detectors are updated, preserving the stability and integrity of the remaining forest, and thus maintaining the model's overall performance more intelligently and consistently.

2.3 Anomaly detection methods based on deep learning

In recent years, deep learning techniques have been widely used in anomaly detection [4], especially autoencoders and generative adversarial networks (GANs), which identify anomalies by reconstructing errors. These methods are effective in detecting complex patterns in data. However, these methods typically require a large amount of data for training and are computationally

demanding. More importantly, they mostly focus on learning spatial features while ignoring the crucial temporal dependencies in streaming data.

2.4 The rise of self-supervised learning in time series anomaly detection

Self-supervised learning [5] provides a new paradigm for solving the problem of scarce abnormal samples by designing pre-training tasks to learn representations from unlabeled data. In particular, in the field of natural language processing (NLP), models such as GPT (Generative Pre-trained Transformer) have learned rich language representations on large-scale corpora through the pre-training task of "predicting the next word." Inspired by this, researchers began to apply similar ideas to time series data, learning the time series evolution pattern of normal data by "predicting future data points." Currently, there is no research on combining this framework with traditional efficient anomaly detection algorithms (such as isolation forests) for power dispatching flow data scenarios. This paper is to integrate GPT-style self-supervised learning with adaptive isolation forests to innovate, capture the time series rules of historical flow data through pre-trained state memory units, and inject the learned representations into the forest detection process, thereby achieving further improvement in detection performance.

In summary, this paper is based on an in-depth analysis of the shortcomings of the isolation forest algorithm for streaming data, and innovates from two dimensions: model update strategy and time series feature mining. It proposes a solution that is both efficient, adaptable, and accurate, filling the gap in existing research in the specific application scenario of power dispatching and control systems.

3 System analysis and methods

3.1 Analysis of the relationship between business, process, and process resource occupancy in the power dispatching control system

With the continued expansion of the dispatching network, the addition of new functions, the upgrade of the dispatching system, and the introduction of demand-side management, the power dispatching control system itself has become increasingly complex. Software errors in the system have become increasingly prominent, and failures caused by software errors have also increased. Based on the construction of the knowledge graph of the smart grid control system in the previous chapter, we can further explain and analyze the relationship between the business, process, and process resource usage of the power dispatching control system. For example, when periodically collecting and recording process data at a telemetry point in a province's power generation, if the difference in values between adjacent moments is greater than a manually set threshold, that is, the data information is abnormal, it is considered that a data jump failure may have occurred.

As shown in Figure 1, the causes of data jumps can be categorized into two main types: one is external factors, such as jumps in the actual data transmitted by a remote station; the other is failures in the local power dispatching and control system, primarily caused by hardware damage and abnormal process operation. Hardware damage is relatively easy to troubleshoot, while failures caused by abnormally running processes are more difficult to detect in a timely manner. Process abnormalities often lead to abnormal hardware resource usage by processes. In the operation of power dispatching and control systems, services and processes are closely intertwined. Figure 2 shows the flow chart of the telemetry table refresh service. When the telemetry table refresh service operates normally, the corresponding processes must run in an orderly and coordinated manner. During this time, the resource usage of each process can be collected. Conversely, abnormal operation of a related process can impact the overall service flow, hindering service operations and even causing failures. Figure 3 illustrates the relationship between services, processes, and process resource usage. Process resource usage can be used to understand the current process operation and, therefore, infer the current service status. In power dispatching and control systems, some services are not only directly affected by the resource usage of related processes, but also indirectly affected by the server's own total resource usage. The reason is that if processes unrelated to the business operation consume excessive server resources, it will affect server performance and thus hinder the business operation. In fact, the resource usage of a process and the total server resource usage are in a total-to-score relationship, but they emphasize different points: the resource usage of a single process mainly reflects the real-time status of the process closely related to the business, which is a direct influencing factor; while the total server resource usage emphasizes the impact of the server's operating status on the business, which is an indirect influencing factor.

Currently, methods for determining the operational status of upper-level services based on process resource usage often rely on a single threshold setting based on expert experience. This approach is highly subjective and fails to fully reflect the relationships between processes associated with the service itself, nor does it reflect the impact of the runtime of processes associated with other services. Online data, however, not only reflects the real-time status of each process during service operation but also implicitly captures the cooperative and competitive relationships between processes. By intelligently analyzing and learning historical data on service process resource usage, and using different attributes as input, defining boundaries in a high-dimensional space that encompass the majority of normal data, and subsequently analyzing and determining whether the operational status of upper-level services is abnormal, this approach is crucial for maintaining the safe and stable operation of the power grid.

3.2 Anomaly detection algorithm for stream data in power dispatching control systems based on adaptive isolation forest

The Isolation Forest Algorithm [5, 6] is a recently proposed and most influential detection algorithm. Its main idea is to randomly select a sample attribute for a data sample space to perform spatial segmentation, obtaining two sub-sample spaces. Then, a

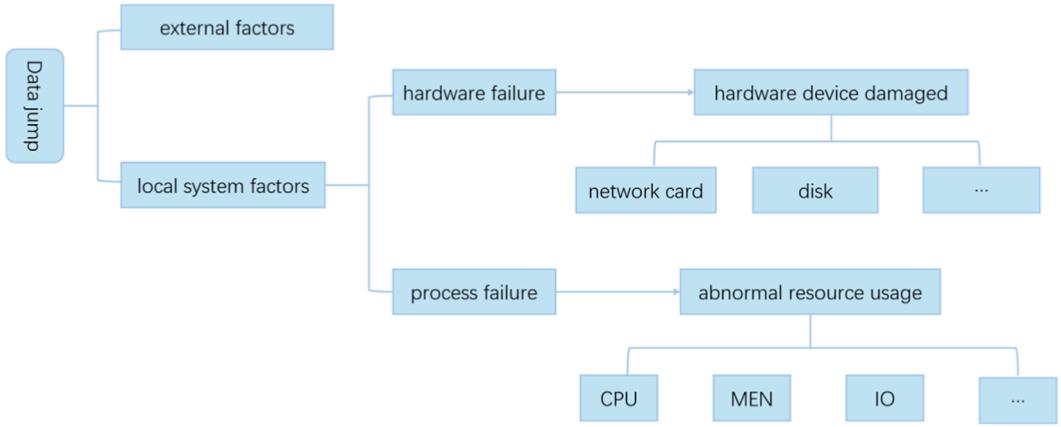

FIGURE 1
Data jump fault analysis.

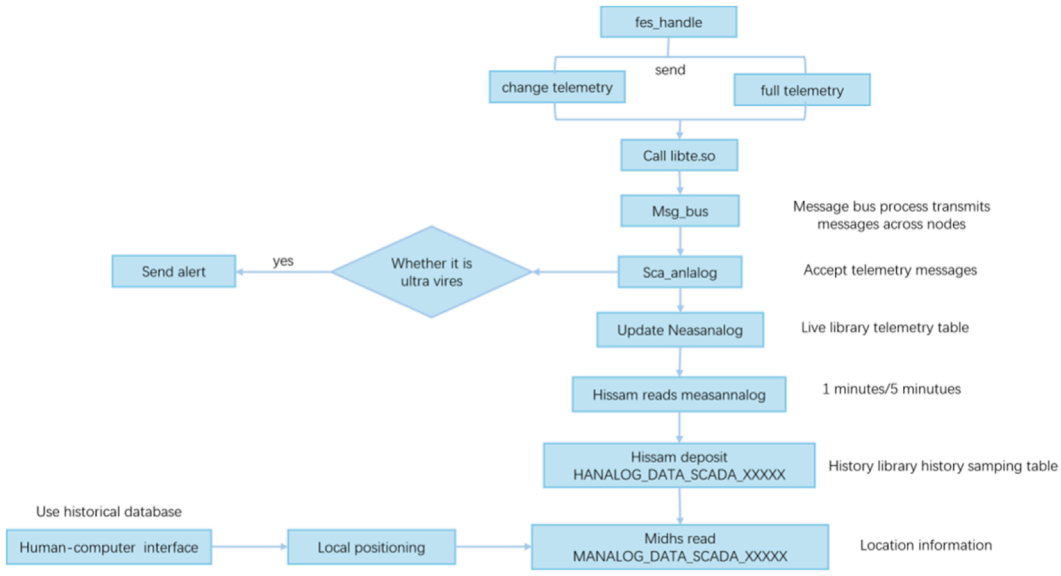

FIGURE 2
Telemetry table refresh business process diagram.

sample attribute is randomly selected to split each sub-sample space until each sub-sample space contains only one type of data point. The partitioning is like building a binary tree, with the root representing the entire sample space and the branches and leaves at the end representing one type of data point. The algorithm is described as follows:

Suppose there is a dataset X and a binary tree T describing the data. It has a set of nodes N, and each node is either N_{ijr} or N_{ijl}, where i represents the number of levels in the tree, j represents the jth node from left to right in the previous level, and r and l distinguish right and left nodes on the same level. In particular, N_0 represents the root node, which contains the data of the entire dataset X.

X contained in a certain layer, randomly select the sample attribute q and its value range space p to divide $X_{(i+1)j^*l}$ and $X_{(i+1)(j^*+1)r}$, corresponding to the node sets $N_{(i+1)j^*l}$ and $N_{(i+1)(j^*+1)r}$. Data less than or equal to p is divided into $N_{(i+1)j^*l}$, and data greater than p is divided into $N_{(i+1)(j^*+1)r}$, where j^* represents the j^* th node from left to right in the i+1th layer. For the dataset, there are shown in Equations 1, 2:

$$X^{(i+1)j^*l} \cup X^{(i+1)(j^*+1)r} = X_{ij} \tag{1}$$

$$X_{(i+1)j^*l} \cap X_{(i+1)(j^*+1)r} = \emptyset \tag{2}$$

When the following situation occurs, a complete binary tree is obtained and the partitioning is completed:

1. The depth of the data tree reaches the set maximum value
2. Node N contains only one data point or the data points it contains are the same

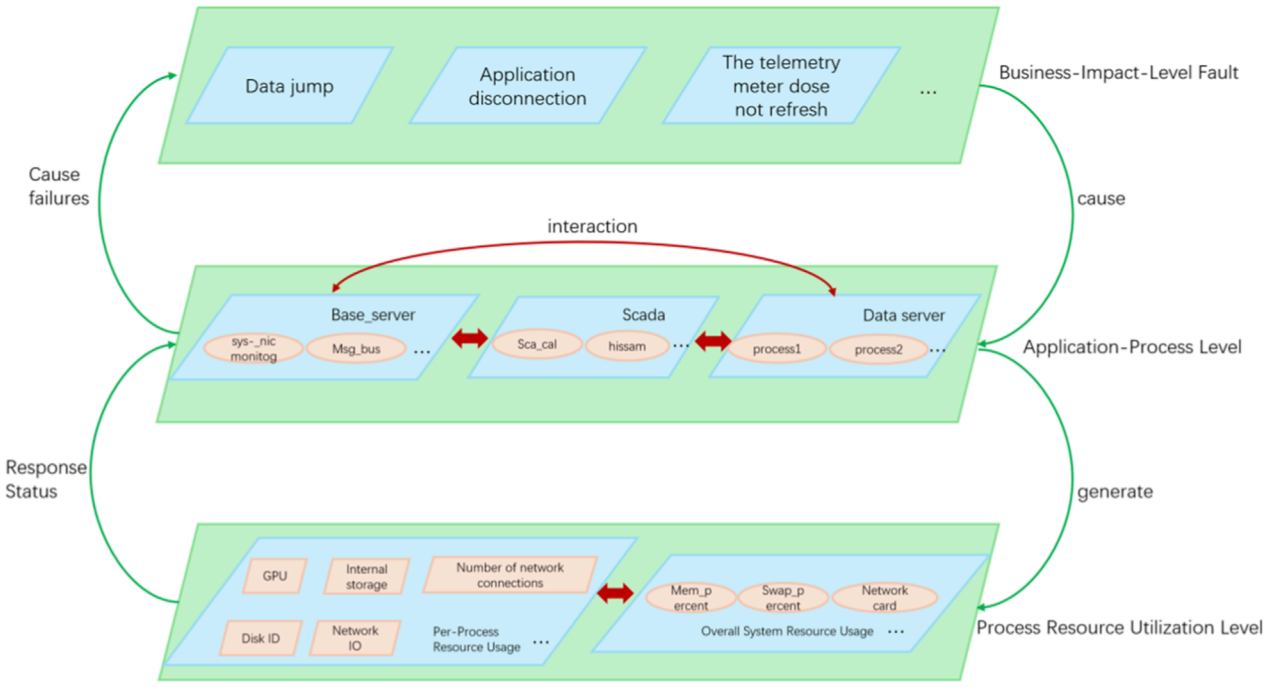

FIGURE 3
Schematic diagram of the relationship between business, process and process resource usage.

The above method is different from existing distance-based and density-based anomaly detection methods. This method does not require distance or density calculation and can better meet the needs of high-efficiency analysis and processing of online anomaly detection.

Streaming data from power dispatching and control systems is characterized by high volume, rapid, and continuous arrival. To detect anomalies in this data, real-time updates of anomaly detectors are necessary to ensure stable performance.

Figure 4 provides an overview of the entire streaming anomaly detection pipeline. Incoming data are first organized into a sliding window of length N, which captures the most recent observations required for window-level anomaly analysis. This window is passed to the anomaly scoring module, where the feature extractor computes the anomaly scores and produces the window-level anomaly rate. To handle the non-stationary nature of power system streams and to remain resilient to concept drift, the framework incorporates an adaptive update mechanism driven by two triggers:

1. An anomaly-rate threshold, where the averaged anomaly score exceeds a predefined value, and
2. A buffer-based triggering strategy, where the update buffer becomes full. These triggers ensure that model updates occur only when necessary, balancing computational efficiency and adaptability.

Integrating the low complexity and high efficiency of the isolation forest algorithm, a novel isolation forest anomaly detection method based on incremental learning with sub-forest progressive updates is proposed for stream data from power dispatching and control systems. Multiple sub-forest anomaly detectors are constructed using isolation trees trained on historical datasets to form a base forest anomaly detector. A sliding window is also created to store the streaming data. Whenever new data enters the window, the oldest data in the window is cleared, implementing a sliding window. A Bernoulli algorithm is then used to determine whether the new data needs to be stored in the buffer.

The base forest anomaly detector determines the anomaly rate of the sliding window. If the anomaly-rate trigger is activated or the buffer becomes full, the system initiates a detector update following the update mechanism described in Figure 4. When the former triggers an update, the updated dataset is the union of the data in the sliding window and the data in the buffer. The sliding window and buffer are cleared, requiring a new sliding window to be rebuilt. When the latter triggers an update, the corresponding updated dataset is the data in the buffer.

Based on the updated dataset, the absolute deviation of the anomaly rate between the sub-forest detector and the base forest anomaly detector is calculated, and sub-forest anomaly detectors with large deviations are removed. Simultaneously, multiple sub-forest anomaly detectors are created based on the updated dataset and added to the base forest anomaly detector to achieve detector update optimization. The process is shown in Figure 5.

This paper proposes an isolation forest-based anomaly detection method for stream data in power dispatching and control systems, which mainly includes five key steps:

Step 1: Sample the data set of the power dispatching control system through a systematic sampling method, construct multiple sub-forest anomaly detectors, and combine the multiple sub-forest anomaly detectors into a base forest anomaly detector.

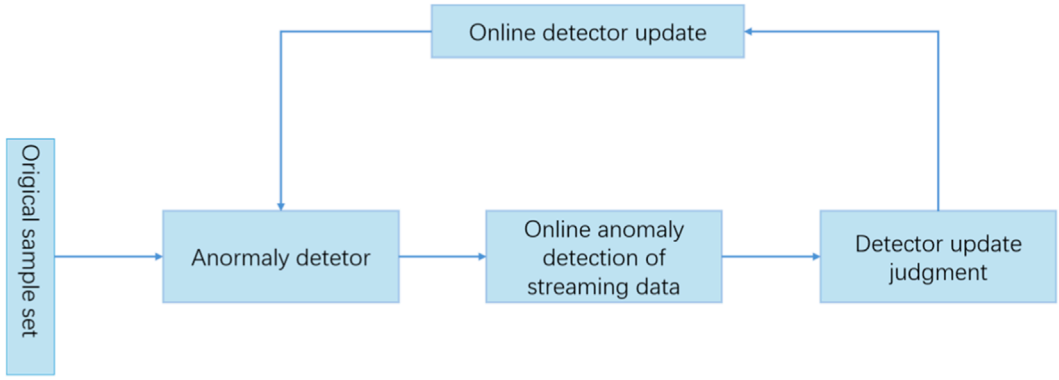

FIGURE 4
Streaming data anomaly detection process.

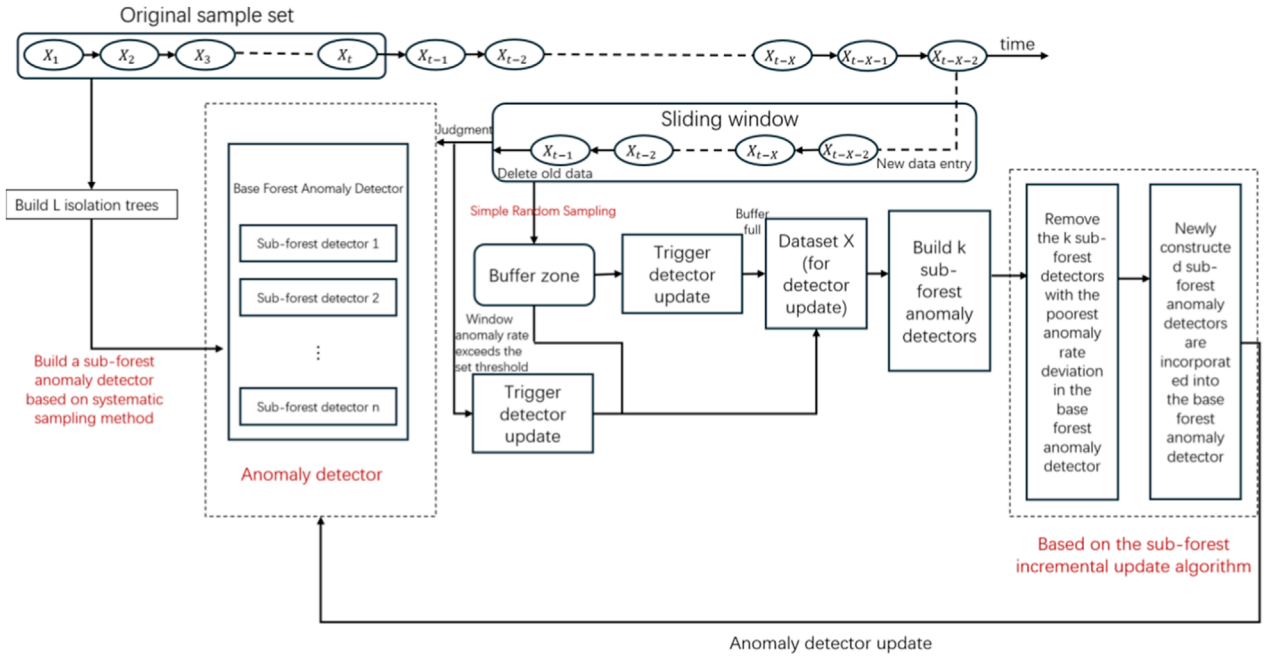

FIGURE 5
Schematic diagram of the anomaly detection algorithm for stream data in the power dispatching control system based on isolation forest.

Specifically, based on the original power dispatching dataset, the isolation forest algorithm is used to construct L isolation trees (where L refers to the number of isolation trees), and the systematic sampling method is used to divide the isolation trees into n groups to construct multiple sub-forest anomaly detectors. The method for all sub-forest anomaly detectors to form the base forest anomaly detector is as follows: collect N power dispatching data samples to form the original power dispatching dataset; uniformly sample Ψ data samples from the N original datasets as training samples for this isolation tree; in each isolation tree sample, two random selections are performed, one is to randomly select a feature, and the other is to randomly select a value within the range of all values of this randomly selected feature, and perform binary partitioning on the sample, dividing samples less than the value to the left side of the node, and samples greater than or equal to the value to the right side of the node, to obtain a split condition and left and right datasets. Then repeat the above process on the left and right datasets respectively until the termination condition is met. There are two termination conditions:

1. The data itself cannot be divided further (it only contains one sample, or all samples are the same);
2. The height of the tree reaches log2(: In the above method, the isolation trees are divided into groups using the systematic sampling method to construct sub-forest anomaly detectors, which are recorded as iForest(1)~iForest(n), where the isolation trees

that make up iForest(i) are numbered as shown in Equation 3:

$$iTree\left(i + k * \frac{L}{n}\right) k = 0, 1, 2, \ldots, \frac{(L-n)}{n} \quad (3)$$

The isolation trees in the n sub-forest anomaly detectors together constitute the base forest anomaly detector.

The algorithm for building the base forest anomaly detector is as follows:

FundamentallForest(Ψ, W,L)
input: Number of training data for each isolation tree, Ψ
number of isolation trees (Tree), L
size of the original sample set L, size of the output set, N
output: Isolation forest (Anomaly detector), IForest

1: Initialize IForest ← {}
2: *h* ← *ceiling(log2 Ψ)*
3: for i ← 1 to *L* do
4: *X* ← *sampleWithReplacement(Ψ, N)*
5: *ITree*←*ITreeTraining(X)*
6: *IForest*←*IForest UITree*
7: end for
8: return *IForest*

Step 2: Use the base forest anomaly detector to determine the anomalies of the data entering the sliding window.

Specifically, the base forest anomaly detector is applied to the streaming data of the sliding window. That is, for each data arriving at the sliding window, its abnormal condition is judged by the base forest anomaly detector. The input of the base forest anomaly detector is the real-time resource occupancy data of the process related to the power dispatching system business, such as process CPU occupancy, memory occupancy, disk IO, network IO, number of threads, number of network connections, etc. (the input features are best organized into a table and placed in the front). The output is a value in the range of (0, 1). The value range indicating that the streaming data is normal is (0, h), and the value range indicating that the streaming data is abnormal is (h, l). The h value represents the anomaly score obtained by using the initial base forest detector obtained by training to calculate the historical data. It can be obtained by taking the quantile based on the abnormal proportion of the historical data as shown in Equation 4:

$$h = -QUARTILE(-F(x), 100 \times (1-c)) \quad (4)$$

Where: $y = QUARTILE(a, b)$ is the quantile function; z = F(x) is the detection function of the base forest detector; X is the training sample set of the isolation tree; c is the proportion of abnormal samples in the training sample set.

Step 3: Sample the stream data entering the sliding window and determine with a certain probability whether it is stored in the buffer; when the sliding window is full of data, determine the abnormality rate of the sliding window data at this time.

Specifically, for data that has just arrived in the sliding window, simple random sampling based on the Bernoulli distribution is performed to determine whether the data needs to be entered into the buffer, thereby filling the buffer. If the sliding window is full, the newly arrived data will replace the data that entered the sliding window the earliest. At the same time, the data anomaly rate in the sliding window at this moment is calculated in real time and recorded as u':

Step 4: When the amount of data in the buffer exceeds the threshold, the update model strategy is triggered according to a smaller update ratio: When the anomaly rate of the sliding window data exceeds the specified threshold, the update model strategy is triggered according to a larger update ratio.

Specifically, the anomaly detector is updated when either of the following two conditions is met:

1. The anomaly rate of the current sliding window data is greater than the anomaly rate threshold. At this time, the dataset X used to update the base forest anomaly detector is the union of the data in the sliding window and the data in the buffer;
2. The data in the buffer is full. At this time, the data set X used to update the base forest detector is the data in the buffer.

Step 5: Based on the updated dataset, calculate the difference between the anomaly rates of each sub-forest anomaly detector and the base forest anomaly detector, remove the sub-forest anomaly detectors with large differences, and construct multiple sub-forest anomaly detectors to supplement them to form a new base forest anomaly detector to achieve the update.

Specifically, the base forest detector and the sub-forest detector are used respectively to calculate the data anomaly rate of the data set X*, denoted as u_{all} and $u_{(i)}$, and set as $r_i i$ the anomaly rate deviation of the th sub-forest anomaly detector iForest(i).

Arrange the anomaly rate deviations of the n sub-forest anomaly detectors in descending order, and take the first k (0 < k < n) as the sub-forest anomaly detectors to be updated;

Use the data in the current sliding window to build k sub-forest anomaly detectors to replace the sub-forest anomaly detectors to be updated. At the same time, update the isolation tree in the base forest detector to complete the update of the base forest detector and clear the data in the sliding window and buffer.

The algorithm pseudocode for steps 2–5 is as follows:
updatingFundamentallForest(X,X $_{buffer}$*,u,u,k,n* $_{per}$*, IForest)*
Input: Current window sample, X′
Buffer sample, Xbuffer
Anomaly rate threshold, u
Abnormality rate of the current window, u′
Number of updated sub-anomaly detectors, k
Number of isolation trees in each group, n_{per}
Current integrated isolation trees (anomaly detectors), with a quantity of m, iForest
Output: Updated integrated isolation trees (anomaly detectors), iForest'

1: Initialize IForest ← {},X*← {},R← {}
2: if X $_{buffer}$ ≥ m then
3: X*←X $_{buffer}$
4: end if
5: if u ′ >u then
6: X*←X ∪ X $_{buffer}$
7: end if
8: fori←-1 to k do
9: IForest $^{*}_{i}$ ← ITree(X *)
10: IForest ′ ← Forest ′∪ IForest′

11: end for
12: u $_{all}$ ←getFailureRate(IForest $_i$,X*)
1. l3: for i ← 1 to m/n $_{per}$ do
14: IForest $_i$ ← $\{ITre\,e_j | j \in [i, i+n_{per}]\}$
15: u(i) ←getFailureRate(IForesti,X*)
16: r ← $|u(i)/u_{all} - 1|$
17: R←R ∪ r $_i$
18: end for
19: Delete the k sub - anomaly detectors with the largest abnormal deviation rate from the IForest in IForest.
20: IForest $'$ ←IForest $'$ ∪ IForest
21: return IForest $'$

3.3 Anomaly detection algorithm for stream data in power dispatching and control systems based on self-supervised learning

The isolation forest-based anomaly detection method for power dispatching control system stream data in the previous section has the characteristics of fast speed and strong real-time performance, but it fails to fully model the serialized information contained in the stream data. At the same time, most of the stream data is negative sample data (i.e., normal data), and only a small amount is positive sample data (abnormal data). Fully mining the knowledge of the transition from negative samples to positive samples is conducive to enhancing the robustness of the system. This section, based on the incremental isolation forest algorithm described in the previous section, further introduces serialized information modeling into the unsupervised anomaly detection process. Specifically, we design a state memory sequence to record the stream data information before the current moment, and enhance the detection performance of the model based on the historical stream data information stored in the state memory unit encoding. At the same time, we use a self-supervised learning framework based on GPT (Generative Pre-trained Transformer) [7] to pre-train the state memory unit and fully mine the information of the transition from negative samples to positive samples. In addition, in order to reduce the continuous similar stream data information, we propose a stream data sampling method based on distance metric to support long-distance memory, reduce storage overhead, and speed up detection.

3.3.1 GPT-based streaming data self-supervised learning framework

Inspired by the recent rise of self-supervised learning language models in the field of natural language processing, we introduce the relevant technologies of the generative language model (GPT) into the self-supervised learning of streaming data, that is, fitting the current streaming data based on historical streaming data [8]. Through this data fitting, self-supervised learning can fully explore the internal laws of streaming data without introducing manual annotation, thereby achieving a good representation of historical streaming data information and improving the performance of subsequent anomaly detection models.

Specifically, for a given stream data sequence $[\ldots, x_{\{t-N\}}, \ldots, x_{\{t-2\}}, x_{\{t-1\}}, x_t, x_{\{t+1\}}, \ldots]$, the standard language model objective function is used to perform maximum likelihood estimation, which is shown in Equation 5.

$$L(\theta) = -\sum_{t=1}^{N} log p_\theta(x_t \mid x_{1:t-1}) \tag{5}$$

Where N represents the window size, θ a neural network with parameters is used to model the conditional probability P, and the optimizer uses the stochastic gradient descent algorithm (SGD). The specific implementation is as follows:

1. The input layer is different from text-based data. Streaming data is often considered to be an infinitely long data sequence and cannot be directly input into the GPT. Therefore, for the time t to be detected, only N+1 sample data containing $X_{\{t\}} = [x_{\{t-N\}}, \ldots, x_{\{t-2\}}, x_{\{t-1\}}, x_{\{t\}}]$ xt are sequentially input. The neural network of the input layer first linearly maps the original streaming data vector, and then adds the interval information from each time step to the time step to be predicted to obtain the output of the input layer $[v\_\{t-N\}, \cdots, v\_\{t-2\}, v\_\{t-1\}, v\_\{t\}]$.
2. GPT-based stream data feature representation

The stream data vector sequence obtained from the input layer V_t is input into the GPT network to obtain the feature vector of the time step data to be predicted h_t. As shown in Figure 6, the GPT network contains L layers of Transformer blocks, and the top layer output is used as the GPT output. Specifically, we use L = 6 layers and Nh = 8 attention heads. These values were chosen based on preliminary experiments to balance the model's capacity for capturing temporal dependencies with computational efficiency, that is $h_i = h_i^L$; the network decomposition of each layer of Transformer blocks is exactly the same, and the mapping of the lth layer can be expressed as $H^l = \text{transformer}_{\text{block}}(H^{l-1}, \theta_l)$, where H^{l-1} is the output of the *l-1th* layer, θ_l represents the parameters of the layer, and the specific calculation process is as shown in Equations 6–9:

$$\text{MultiHead}(Q,K,V) = \text{Concat}(\text{head}_1, \text{head}_2, \ldots, \text{head}_{N_h}) \tag{6}$$

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \tag{7}$$

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}} + \text{mask}\right)V \tag{8}$$

$$transformerblock(H) = FNN(MultiHead(H,H,H)) \tag{9}$$

where $headi \in R^{N \times dh}$ represents the result of calculation of the i-th head in the multi-head attention module MultiHead; Concat concatenates the features of the last dimension of each head, and the result satisfies $Concat(head1, head2, \ldots, headNh) \in R^{N \times d_{mh}}$, where $d_{mh} = d_h \times N_h$; mask is the mask matrix used to calculate the time step h_i^l, only considering $[h_{t-N}^{l-1}, \cdots, h_i^{l-1}]$, that is $\text{mask}_{ij} = 0$. When $i \geq j$, $\text{mask}_{ij} = -\infty$.

3. Probability estimation of data to be predicted

Based on the historical stream data feature representation obtained by the Transformer decoder, the probability of the stream data at the target time is calculated as shown in Equation 10:

$$P(x_i \mid x_{i-k}, \ldots, x_{i-1}; \theta) = \prod_{j=1,\ldots,N} P_n(x_{ij} - x'_{ij}) \tag{10}$$

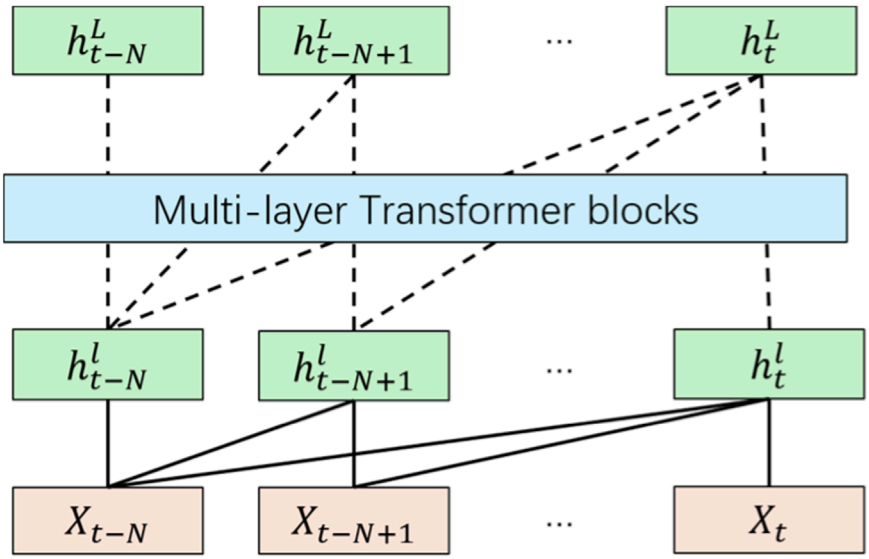

FIGURE 6
GPT basic network structure.

After the probability obtained in this way is substituted into the language model objective function, the model optimization can be achieved according to the objective function.

3.3.2 Stream data sampling based on distance metric

We choose the L2 norm as the distance metric for sampling due to its computational efficiency and its ability to effectively capture Euclidean distances between data points. The L2 norm is commonly employed in anomaly detection tasks because it highlights the differences in the feature space, which is essential for distinguishing anomalous data in high-dimensional streaming environments.

Actual sampled stream data contains numerous consecutive, similar data points, resulting in a significant amount of redundant information. Calculating all of these points would slow down model detection and introduce unnecessary, irrelevant information, hindering the model's ability to capture useful information. To address this issue, we designed a stream data sampling method based on distance metrics to reduce redundant information. Specifically, given the original stream data $[x_1, x_2, \ldots, x_n]$, we traverse it from front to back and sample only those points that meet the following requirements Equation 11:

$$\| x_i - x_{latest} \|_2 > \varepsilon \tag{11}$$

represents x_{latest} the most recent sampling, ε and the minimum distance interval that sampling needs to meet is represented by, thus obtaining the sampled flow data $[\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_n]$.

3.3.3 Streaming data anomaly detection method based on self-supervised learning

Figure 7 provides an overview of the proposed self-supervised learning framework used for pre-training the GPT-based representation model.

The core idea is to leverage historical streaming data to learn temporal patterns without relying on manual labeling. To this end, the method constructs a state memory that stores a representative subset of the historical data. This memory is populated using a distance-based sampling strategy, ensuring that the stored sequences are diverse and preserve long-term temporal variation.

During pre-training, a sequence $X = [x_1, x_2, \ldots, x_N]$ is fed into the model, which performs a next-step prediction task following the standard autoregressive language modeling objective. The GPT network learns to predict each data point conditioned on its preceding context, thereby capturing temporal dependencies and normal operating patterns of the system. The output of the top-layer Transformer block serves as the temporal representation h_t, which is later used as the feature input for anomaly detection.

Once pre-training is completed, the learned parameters are frozen and deployed in the online detection pipeline. This separation between offline representation learning and online detection ensures both efficiency and stability, as shown in Figure 7. The self-supervised framework enables the model to generalize to unseen patterns and forms the foundation for robust anomaly detection in the streaming environment.

3.3.3.1 State memory unit construction

In order to introduce the flow data information before the current time step into the anomaly detection algorithm of the power dispatching control system based on isolation forest proposed in Section 3.2, the self-supervised learning framework described in Section 3.3.2 is used to construct a state memory unit based on the GPT network h_t: $F(X_t, \theta^*)$ The specific process is as follows:

1. Data sampling: For each moment, first sample a stream data sequence of length N starting from that moment $[x_{t-N+1}, x_{t-N+2}, \cdots, x_t]$, and further sample the input of the state memory unit at that moment based on the distance

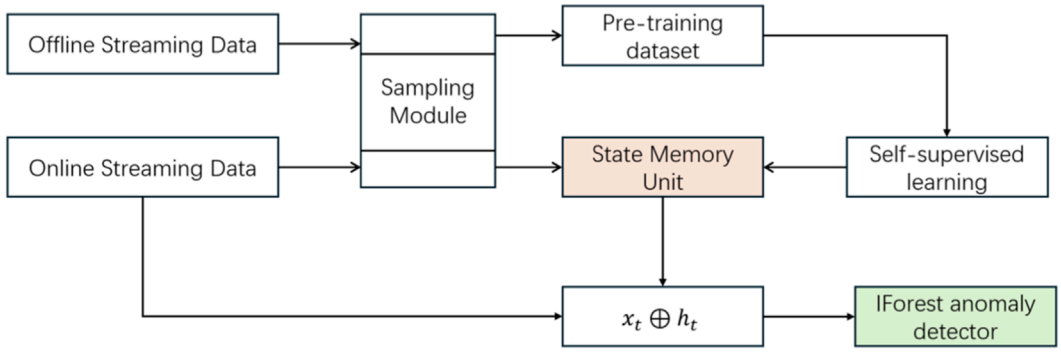

FIGURE 7
Streaming data anomaly detection framework based on self-supervised learning.

metric sampling technology $Xt = [x_1^t, x_2^t, \ldots, x_{Nt}^t]$; for the pre-training stage, first collect enough offline historical stream data, traverse each moment t, and sample the pre-training data set $\{\ldots, X_{t-1}, X_t, X_{t+1}, \ldots\}$

2. Pre-training: Based on the pre-training dataset $\{\ldots, X_{t-1}, X_t, X_{t+1}, \ldots\}$, the parameters of the state memory unit are trained using the self-supervised learning method described in Section 3.4.1 until convergence is reached, and the model parameters of the state memory unit are saved θ^*;
3. Time series feature calculation: After training is completed, for a given stream data point to be detected x_t, the same sampling strategy is used to obtain its historical stream data sequence X_t, which is input into the state memory unit and the historical stream data information vector is output h_t.

The parameters of the state memory unit θ^* are completely obtained by pre-training based on self-supervised learning, and no parameter updates are performed when training on streaming anomaly detection data.

3.3.3.2 Anomaly detection

Based on the trained state memory unit, the input of the original Section 3.2 for anomaly detection of stream data in the power dispatching control system based on isolation forest x_t is replaced by $x_t \oplus h_t$, and the anomaly detection process and the update of the base anomaly detector are consistent with Section 3.2.

4 Experimental results and comparative analysis

The dataset used in this study consists of process resource usage data collected from a computer running Python and other business processes. It includes 18 dimensions, mainly CPU usage (%), memory usage (MB), and IO read/write rate (MB/s). For the anomaly detection experiments, the dataset is divided into three subsets: the training set, the streaming update set, and the testing set. The training set contains 1,000 samples without anomaly labels for model initialization; the streaming update set contains 3,100 samples without anomaly labels to simulate stream data input and model updating; the testing set contains 2,472 samples with anomaly labels, among which 918 samples are labeled as anomalous. Anomalies are defined as Python running data exceeding 2 GB, while idle or non-computing periods are labeled as normal. This threshold is determined based on historical data distribution and empirical observation. The receiver operating characteristic curve (ROC) is often used to describe the performance of the anomaly detection algorithm [10, 11]. An effective anomaly detection method needs to maintain a high recall rate and precision rate, and the balance between the two can be described by the ROC curve. When evaluating an algorithm, a quantitative metric is often needed to measure anomaly detection performance. This can be quantified using the area under the receiver operating characteristic (ROC) curve (AUC).

Table 1 summarizes the key statistics of the datasets used in the experiments. Each dataset contains 18 feature dimensions, including CPU usage, memory usage, and IO read/write rates. The "Sample Size" column indicates the total number of samples in each subset, and "Anomalous Samples" shows the number of samples labeled as anomalous (none in the training and streaming update sets, 918 in the testing set). The statistical columns ("Mean," "Std Dev," "Max," "Min") represent the average, standard deviation, maximum, and minimum values across all dimensions and all samples in the respective dataset. All features were normalized to the range [0,1] prior to input. This table provides readers with an overview of dataset composition, variability, and anomaly distribution.

To ensure consistency for model training and testing, all features are normalized to the range of 0–1 before input. The streaming data is input in its original temporal order, and sliding windows of 64 samples are used to generate input features, ensuring the real-time characteristics of the stream data. The update strategy includes updating sub-detectors when the cache is full and adjusting the update rate when the estimated anomaly rate of the sliding window exceeds a preset threshold, addressing concept drift and clustered anomalies.

4.1 Selection of sampling size and integration scale

The ensemble size and sliding sampling window size affect the algorithm's AUC performance, so it is important to select an appropriate combination to ensure optimal model AUC performance. The ensemble size range is (20, 40, 60, 80, 100, 120),

TABLE 1 Detailed dataset statistics.

Dataset	Sample size	Anomalous samples	Dimensions	Mean	Std DeV	Max	Min
Training set	1,000	0	18	0.52	0.15	0.95	0.01
Streaming update set	3,100	0	18	0.50	0.16	0.98	0.02
Testing set	2,472	918	18	0.54	0.18	1.00	0.00

TABLE 2 AUC values under different integration scales and window sizes.

Scale	Sliding sampling window size				
	64	128	256	512	1,024
20	0.8058	0.7358	0.6348	0.5180	0.5159
40	0.8096	0.6769	0.6832	0.5216	0.4835
60	0.8465	0.7285	0.6070	0.6175	0.4646
80	0.8416	0.7032	0.6560	0.5358	0.3672
100	0.8092	0.7475	0.5889	0.5540	0.5145
120	0.7779	0.7039	0.6380	0.5791	0.4363

and the sliding sampling window size range is {64, 128.256, 512, 1,024}. The actual values of these two parameters depend on the test data, so when performing this selection experiment, no isolation tree updates are performed, meaning there's no need to set an update ratio.

Since no isolation tree updates are performed, the experimental results of the algorithm in Ref. [9] are consistent with those of this paper, that is, the integration scale and sliding window sampling size are the same. The same integration scale and sliding window sampling size are selected for subsequent experimental comparisons. The purpose is to control variables and then explore the different performances of the two algorithms when the update ratio changes, and compare them. By inputting the data set in sequence to simulate the characteristics of the streaming data, the experimental results obtained are shown in Table 2. The results show that for this data set, the algorithm has a higher AUC value when the integration scale is 60 and the sliding sampling window size is 64. Therefore, these two parameters are used for subsequent experiments.

4.2 Algorithm comparative analysis

4.2.1 Performance comparison

The selection of the update rate parameter requires consideration of specific circumstances. There are two scenarios that trigger detector updates: First, an update is triggered by a full buffer. In this case, the estimated anomaly rate of the sliding window is below a pre-set threshold, so a larger update rate is not required. Second, an update is triggered by an estimated anomaly rate of the sliding window exceeding a threshold. This scenario can occur in two ways: one is that the data sample is normal, but a larger update is required due to concept drift; the other is that the anomaly rate is high due to clustered anomalies, but the detector is still usable, so the update rate should be minimized.

In this paper, the update ratio is set to be less than 0.5, and some discrete points are selected for simulation experiments. The update ratio set is (0.1, 0.2, 0.3, 0.4). Due to the presence of Bernoulli randomness when selecting data, in order to obtain more objective evaluation results, as shown in Table 3, in this experiment, the method proposed in this paper and the method proposed in the literature [9] were respectively repeated 20 times at the same update ratio to obtain the corresponding AUC values. The statistics were calculated and the average AUC value was recorded as the experimental result. This additional information helps to ensure that the improvements in performance are statistically significant and not influenced by random fluctuations. Since the flow data anomaly detection has the requirement of real-time performance, a method similar to the above AUC value test was adopted to compare the detection speed of each method. The detection speed results are shown in Tables 3, 4.

The proposed method outperforms traditional anomaly detection algorithms (e.g., Isolation Forest and Autoencoders) in several key aspects. First, our adaptive isolation forest mechanism addresses the performance degradation caused by concept drift, which is a significant issue for static algorithms. As shown in Table 3, our method demonstrates a 34.12% improvement in AUC compared to the baseline algorithm, especially under conditions of high concept drift (update ratio = 0.4). Second, the integration of self-supervised learning based on GPT allows our method to capture the temporal dependencies within the data, which traditional methods like Isolation Forest fail to do. This is particularly crucial in streaming data environments, where time-series patterns play a major role in anomaly detection. Our results show a significant increase in the detection performance, with AUC values improving by up to 39.12% compared to existing methods.

4.2.2 Ablation experiments

An ablation study involves removing some improved features from a relevant model or algorithm to verify their necessity. This paper presents an adaptive isolation forest algorithm and a self-supervised learning algorithm for anomaly detection in stream data of a power dispatching control system. Ablation experiments were conducted to verify the effectiveness of the improvements made to each submodule. Each set of ablation experiments was conducted at different update ratios. The result is as shown in the Table 5.

TABLE 3 Comparison of AUC values under different methods.

Method comparison	Update ratio/AUC value comparison							
	0.1		0.2		0.3		0.4	
Isolation forest method for raw stream data	0.6869		0.7661		0.7636		0.6356	
Adaptive isolation forest	0.7110	3.193%	0.7667	0.131%	0.7618	−0.262%	0.8534	34.119%
Based on self-supervised learning	0.7515	8.999%	0.7965	3.916%	0.8454	10.602%	0.8843	38.994%

TABLE 4 Comparison of detection speed under different methods (seconds/1,000 sampling points).

Method comparison	Update ratio/AUC value comparison			
	0.1	0.2	0.3	0.4
Isolation forest method for raw stream data	0.3454	0.3542	0.3771	0.4016
The method proposed in Section 3.3 of this paper (adaptive isolation forest)	0.3612	0.3723	0.3886	0.4126
The method proposed in Section 3.4 of this article (based on self-supervised learning)	0.7341	0.7615	0.7892	0.8214

TABLE 5 Ablation experiment of stream data anomaly detection based on isolation forest.

Method	Update ratio/AUC value comparison t-test			
	0.1	0.2	0.3	0.4
Isolation forest method for raw stream data	0.6890	0.7661	0.7636	0.6356
The method proposed in Section 3.3 of this paper (adaptive isolation forest)	0.7110	0.7667	0.7618	0.8534
-Adaptive	0.6145	0.6525	0.6618	0.6123
-Forest	0.5511	0.5631	0.5694	0.5661

Among them, the following ablation is performed on the adaptive isolation forest algorithm proposed in Section 3.2 of this article:

1. Adaptive: Instead of using incremental learning, all sub-anomaly detectors are learned using only the data in the current cache. All sub-anomaly detectors are replaced each time the model is updated.
2. Forest: Only one sub-anomaly detector is used without integrating multiple sub-anomaly detectors;

For the anomaly detection algorithm based on self-supervised learning proposed in Section 3.3 of this paper, the following ablation is adopted:

1. Self-supervised learning: Instead of using the self-supervised learning state unit to encode the historical stream data information into ht, we simply concatenate the sampled historical stream data vectors as ht:
2. Distance-based sampling: During pre-training and testing, no distance-based sampling is performed to process the input data of the state memory unit.

The t-test results indicate that the improvements introduced by the 'Adaptive' and 'Self-supervision' modules are statistically significant. This confirms that each module contributes meaningfully to the overall performance, and the improvements are not due to random variations. The result is as shown in the Table 6.

4.3 Experimental results analysis

Analysis of the experimental results shows that the experimental results of the two methods under different update ratios are different. Although there are several other methods in the literature for anomaly detection, we chose to compare our method with the one presented in [9] because it is the most relevant to our approach and operates under similar conditions. Methods like autoencoders and GANs, while promising, were not included in the comparison for this study, but we plan to include them in future work for a more comprehensive evaluation. Compared with the method proposed in Ref. [9], the AUC value of the method proposed in this paper is significantly improved when the update ratio is 0.1 and 0.4, and is relatively close when the update ratio is 0.2 and 0.3. In addition, the

TABLE 6 Ablation experiment of stream data anomaly detection based on self-supervised learning.

Method	Update ratio/AUC value comparison t-test			
	0.1	0.2	0.3	0.4
Isolation forest method for raw stream data	0.6890	0.7661	0.7636	0.6356
Self-supervised learning methods	0.7515	0.7965	0.8454	0.8843
-Self-supervision	0.7110	0.7867	0.8018	0.7134
-Distance-based sampling	0.7312	0.7893	0.8332	0.8416

AUC values of the two methods proposed in Sections 3.3 and 3.4 of this paper at 0.4 are improved by 34.27% and 39.12% respectively compared with the method proposed in Ref. [9]. Performance comparison experiments show that the power dispatching control system flow data anomaly detection algorithm based on self-supervised learning proposed in this paper has better comprehensive performance.

In terms of detection speed, the adaptive isolation forest algorithm proposed in Section 3.2 of this paper is comparable to the baseline algorithm. However, the self-supervised learning-based approach proposed in Section 3.3 is relatively slow. This is primarily due to the significant computational time required by the neural network used in this approach during forward computation. Therefore, in practical applications, this approach places higher demands on the real-time performance of other system components. Considering the actual margin required for real-time performance of streaming data in power dispatching and control systems, the speed disadvantage of the self-supervised learning-based approach remains acceptable.

The ablation experiment results based on the adaptive isolation forest algorithm proposed in Section 3.2 of this paper show that adaptively updating the model is of great significance to the proposed method. Without the proposed adaptive update mechanism, the model detection performance degrades significantly. At the same time, ensemble learning plays a fundamental role in anomaly detection. Compared with the adaptive update mechanism, the performance degradation is more obvious when the ensemble learning algorithm is not used.

The ablation experiments based on the self-supervised learning algorithm presented in Section 3.3 of this paper show that the temporal features obtained by self-supervised learning not only improve the model's detection performance but also provide more stable performance under different update rates. Furthermore, ablation experiments comparing "self-supervised learning" with "adaptive isolation forest" demonstrate that temporal information is crucial for anomaly detection in streaming data, improving detection performance even without self-supervised learning. Distance-based sampling also significantly improves the modeling of temporal features and patterns in streaming data by removing redundant information from the data stream, specifically sampling points that have not changed significantly from the previous moment.

5 Conclusion

To address the practical needs of anomaly detection in power grid dispatching services, this paper proposes an adaptive isolation forest-based anomaly detection algorithm for streaming data in power dispatching and control systems. Furthermore, a self-supervised learning framework is introduced to further improve model performance. Taking into account the characteristics of streaming data in power dispatching and control systems, a new isolation forest anomaly detector update strategy is proposed. This strategy discards sub-forest anomaly detectors with large anomaly rate deviations and replaces them with new sub-forest anomaly detectors. This addresses the issue of overall performance degradation of anomaly detectors caused by random updates and improves the algorithm's anomaly detection performance. The self-supervised learning framework employs a deep learning neural network and training techniques similar to the GPT pre-trained language model to model the temporal characteristics and patterns in streaming data, further improving the model's detection performance. Training and testing on a simulated streaming dataset with temporal characteristics demonstrate the proposed method's superiority in comprehensive anomaly detection performance, including recall and precision, as well as the effectiveness of its key technical features. In view of the current situation that there are too few abnormal samples and the types of abnormalities are not rich, based on an in-depth analysis of the various business-related processes and their topological relationships in the power dispatching system, further improving the comprehensive performance of the anomaly detection method and expanding its applicability through data collection and accumulation will be one of the key research points in the future.

Data availability statement

The original contributions presented in the study are included in the article/supplementary material, further inquiries can be directed to the corresponding author.

Author contributions

TX: Investigation, Writing – original draft, Funding acquisition, Methodology, Project administration, Supervision. BL: Investigation, Writing – review and editing, Methodology. YZ: Data curation, Investigation, Writing – review and editing. HW: Methodology, Supervision, Writing – review and editing. KX: Formal Analysis, Methodology, Writing – review and editing. JH: Formal Analysis, Project administration, Writing – review and editing.

Funding

The authors declare that no financial support was received for the research and/or publication of this article.

Conflict of interest

Authors TX, BL, YZ, HW, KX, and JH were employed by Information Center of Guangdong Power Grid Co., Ltd.

Generative AI statement

The authors declare that no Generative AI was used in the creation of this manuscript.

Any alternative text (alt text) provided alongside figures in this article has been generated by Frontiers with the support of artificial intelligence and reasonable efforts have been made to ensure accuracy, including review by the authors wherever possible. If you identify any issues, please contact us.

Publisher's note

All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

References

1. Liu FT, Ting KM, Zhou ZH. Isolation forest. In: *Data mining and knowledge discovery*. Springer US (2019). p. 1–36.

2. Hariri S, Kind MC, Brunner RJ. Extended isolation forest. *IEEE Trans Knowledge Data Eng* (2021) 33(4):1479–89. doi:10.1109/tkde.2019.2947676

3. Chen Y, Zhou L, Pei S, Yu Z, Chen Y, Liu X, et al. SRF: self-supervised random forest for anomaly detection in streaming data. *IEEE Trans Neural Networks Learn Syst* (2022).

4. Li Z, Zhao Y, Hu Q, He N, Yao L. EIF: an extended isolation forest for anomaly detection in big data. In: *2020 international joint conference on neural networks (IJCNN)*. IEEE (2020). p. 1–8.

5. Doshi K, Yilmaz Y. Continual learning for anomaly detection in streaming data. *Eng Appl Artif Intelligence* (2023) 123:106439. doi:10.1109/ACCESS.2024.3377690

6. Lu J, Zhang W, Hamzei M, Jafari N. The applications of machine learning mechanisms in the compositions of internet of things services: a systematic study, current progress, and future research agenda. *Eng Appl Artif Intelligence* (2025) 147:110345. doi:10.1016/j.engappai.2025.110345

7. Zhang H, Song Y, Wang Y. A self-supervised learning framework for anomaly detection in industrial IoT. *IEEE Internet Things J* (2021) 8(15):12345–56.

8. Gu X, Akoglu L, Rinaldo A. Statistical analysis of anomaly detection in attributed streams. In: *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery and data mining* (2019). p. 1897–907.

9. Wang H, Bai J, Wang J, Wang Z. An adaptive ensemble method for anomaly detection in power grid streaming data. *IEEE Trans Power Syst* (2022) 37(2):1542–53.

10. Pang G, Shen C, Cao L, van den Hengel A. Deep learning for anomaly detection: a review. *ACM Comput Surv (Csur)* (2021) 54(2):1–38. doi:10.1145/3439950

11. Zhao Y, Nasrullah Z, Li Z. PyOD: a python toolbox for scalable outlier detection. *J Machine Learn Res* (2019) 20(96):1–7.

12. Lu JZ, Wang CL, Su J, Ding K, Liu X. An adversarial example defense algorithm for intelligent driving. *IEEE Netw* (2024) 38:98–105. doi:10.1109/mnet.2024.3392582