Anomaly Diagnosis with Siamese Discrepancy Networks in Distributed Cloud Databases

Lingsen Yan[1], Bolong Zheng[1,*], Junjie Qing[1], Wenlong You[1], Tingyang Chen[1], Zhi Xu[2], Shuncheng Liu[2], Kai Zeng[2], Tao Ye[2], Xiaofang Zhou[3]
[1]*Huazhong University of Science and Technology, Wuhan, China*
[2]*Huawei, Shenzhen, China*
[3]*Hong Kong University of Science and Technology, Hong Kong, China*

2025 IEEE 41st International Conference on Data Engineering (ICDE) | 979-8-3315-3603-9/25/$31.00 ©2025 IEEE | DOI: 10.1109/ICDE65448.2025.00302

***Abstract*—Anomaly diagnosis is a fundamental problem in operation and maintenance of distributed cloud databases. Existing deep learning based methods solve this problem by classifying the anomalies with different root causes. However, since anomalies seldom occur, and anomalies with the same root cause may exhibit significantly different behaviors across different cloud database clusters, existing methods often lack sufficient training data, and they cannot generalize well from some clusters to others. Therefore, we take both anomaly and normal data into consideration, based on an observation that the discrepancy between the anomaly and normal data is relatively consistent compared to the behaviours of anomalies themselves. We design a Siamese Discrepancy Network (SDN) to learn representations of such discrepancy under the case that only a small amount of training data is available. In addition, a discrepancy-based diagnosis paradigm is proposed to construct training data for SDN and diagnose based on representations of discrepancy learned by SDN. Finally, we develop an anomaly interpretation method based on SDN, which accurately locates the symptom KPIs and root cause KPIs. Extensive experiments are conducted on both synthetic and real-world datasets. The experimental results demonstrate that the proposed method outperforms existing methods with respect to anomaly diagnosis and anomaly interpretation. In particular, the anomaly diagnosis framework has already been applied in Huawei's GaussDB (DWS) system.**

***Index Terms*—anomaly diagnosis, distributed cloud database, time series, deep learning.**

I. Introduction

With the explosion of data and the growth of business scale, database systems have evolved from single-machine setups to distributed cloud architectures, leading to increasingly complex underlying designs. In recent years, cloud database systems are rapidly expanding as a critical part of the infrastructure services provided by major cloud vendors such as Huawei Cloud, Amazon AWS, Microsoft Azure, Alibaba Cloud, etc. Due to the complexity of the architecture, cloud database systems are faced with a wide variety of complex performance anomalies. For example, issues such as I/O-intensive task overload and data skew in distributed nodes can cause severe performance degradation. If these issues are not promptly identified and resolved, they may even lead to service crash, resulting in unpredictable loss for users. This is especially critical in cloud databases, where service providers must strive to guarantee service quality.

Bolong Zheng is the corresponding author.

To detect anomalies, extensive studies [1]–[5] are proposed to determine whether the system is currently anomalous by analyzing the time series of Key Performance Indicators (KPIs), such as CPU Utilization, I/O Utilization, Memory Usage, Active Sessions and etc. However, anomaly detection alone is insufficient; it is essential to identify the underlying root cause in order to effectively troubleshoot the anomaly. For most DBAs, root cause analysis is particularly challenging. Vast amounts of data of KPIs are collected over time. Analyzing these data manually is overly time-consuming and labor-intensive. Therefore, automated methods are developed, some of which locate root cause KPIs [6]–[12] or SQLs [13] and others identify root cause types [14]–[19]. In this study, we focus on identifying the root cause type, because it can provide direct diagnosis results to DBAs.

Unfortunately, existing studies [14]–[16], [20] on anomaly diagnosis in database systems are unable to adequately capture complex patterns. Most of them heavily rely on expert knowledge. They summarize prior features or rules of anomalies that can not fully cover various patterns of anomalies. There are also deep learning based studies [17]–[19] on anomaly diagnosis in general software systems. However, these studies do not consider the characteristics of database anomalies and most of them can not provide interpretations of the reported anomalies.

Huawei Cloud GaussDB (DWS) [21] is a distributed database product designed to handle large-scale datasets and high-concurrency queries. With its superior performance and flexible deployment models, GaussDB (DWS) is widely used in mission-critical business systems and big data analytics scenarios across applications, such as finance, telecommunications, and manufacturing. In such a system, DBAs usually spend extensive time on anomaly diagnosis. Based on feedback from DBAs and analysis of historical cases, we summarize the challenges as follows:

Challenge 1: *There are many anomaly root causes in distributed cloud databases, while the number of cases collected for each root cause is limited.* Due to the complexity of the architecture, cloud database systems are faced with a wide variety of complex anomalies. While vast amounts of monitoring data are generated daily, anomalies are rare. Consequently, the historical cases available for analysis are scarce. Additionally, it is extremely costly for DBAs to identify

2375-026X/25/$31.00 ©2025 IEEE
DOI 10.1109/ICDE65448.2025.00302

Authorized licensed use limited to: University of Electronic Science and Tech of China. Downloaded on October 13,2026 at 12:18:11 UTC from IEEE Xplore. Restrictions apply.

anomalies and label their root causes manually. Therefore, labeled anomaly cases are insufficient, which limits the use of general deep learning-based methods.

Challenge 2: *The effects of anomalies caused by the same root cause vary across different clusters in distributed cloud databases.* Each cluster has different settings in terms of node counts and resource configurations, resulting in variations in KPI behaviors, which leads to poor generalization of existing methods. For example, in cases of CPU overload anomalies caused by highly concurrent tasks, clusters with varying configurations can support different levels of concurrency capacity. The same concurrency related metrics may correspond to different anomaly states across clusters, meaning that "highly concurrent" is a relative description.

Challenge 3: *KPIs exhibit diverse anomaly patterns that are difficult to extract.* Complex components of distributed cloud database systems lead to anomaly with various patterns. Usually anomalies are presented in the trend, level and correlations of KPIs. Most existing studies [14]–[16], [20] focus on only part of them, which limits the performance of diagnosis.

Through analysis on this problem, we have the following observation. We find that while anomaly data is scarce, there is an abundance of normal data, which also provides valuable information for diagnosing anomalies. In anomaly diagnosis tasks, DBAs can usually identify the root cause of an anomaly even with few samples. This ability is primarily attributed to their capability to learn through comparison. DBAs are usually well-acquainted with the patterns of normal data, enabling them to quickly diagnose anomalies by contrasting the discrepancy between anomaly and normal data. Anomalies can be viewed as the combination of anomaly events and normal behaviors, and in other words, anomalies deviate with the normal patterns. Although different clusters may exhibit different normal behaviors, extracting the discrepancy between normal and anomaly data can reveal consistent anomaly patterns. This observation not only helps enhance diagnosis performance but also improves the generalization capability across clusters.

To exploit the above observation, we model the discrepancy between anomaly and normal data, based on which the root causes are identified, instead of only focusing on anomaly data in existing methods. As deep learning techniques have demonstrated powerful capability in representation learning, outperforming traditional methods by extracting various features, we develop a deep learning-based method to learn representations of discrepancy and diagnose anomalies. Specifically, we propose a Siamese Discrepancy Network (SDN) to learn representations of discrepancy effectively, and a discrepancy-based diagnosis paradigm to construct training data for SDN and diagnose anomalies based on representations of discrepancy learned by SDN.

SDN adopts a Siamese architecture, which contains three discrepancy networks with shared weights. In each discrepancy network, there are two feature extraction components with shared weights to extract features of anomaly and normal data respectively and a discrepancy representation component to aggregate features to generate discrepancy representations. In each feature extraction component, we design a Multi-scale Trend CNN (MTCN) and a Hierarchical GCN (HGCN) tailored to database anomalies to extract trend, level, and correlation features (Challenge 3). In addition, we develop a discrepancy-based diagnosis paradigm. In the paradigm, samples are paired to construct training samples. SDN is then trained to learn representations of discrepancy using Triplet Loss [22]. Finally, anomalies are diagnosed based on representations of discrepancy. By introducing normal samples and utilizing the Triplet-based training strategy, the data volume is significantly augmented, which effectively alleviates overfitting of SDN (Challenge 1). Furthermore, since discrepancy patterns of anomalies are consistent across clusters, the discrepancy-based diagnosis has better generalization performance across clusters (Challenge 2). Additionally, benefiting from the learning of discrepancy representations, we design a method for anomaly interpretation based on SDN enabling the localization of symptom KPIs and root cause KPIs.

The main contributions are summarized as follows:

- We propose a discrepancy-based diagnosis paradigm tailored to anomaly diagnosis in distributed cloud databases. The paradigm enables training sample construction, representation learning of discrepancy and online diagnosis. It addresses the challenges of limited historical cases and poor generalization performance across clusters.
- We design a novel deep network SDN to learn representations of discrepancy. Specifically, it takes both anomaly and normal data as input and we design MTCN and HGCN to enhance feature extraction capability based on the characteristics of anomalies of database systems.
- We develop an anomaly interpretation method based on SDN which effectively locates symptom KPIs and root cause KPIs, providing sufficient clues for DBAs to understand and troubleshoot the anomaly.
- To evaluate our method, we conduct extensive experiments on a real world dataset collected on Huawei Cloud GaussDB (DWS) and a public synthetic dataset, which demonstrates the superior performance of our method in anomaly diagnosis and anomaly interpretation.

II. Background and Preliminary

A. Distributed Cloud Database

Huawei Cloud GaussDB (DWS) supports distributed transactions in a cluster-based architecture and ensures high availability through a primary-backup design. A GaussDB (DWS) cluster primarily consists of three types of nodes: Coordinator Nodes (CNs), Global Transaction Managers (GTMs), and Data Nodes (DNs). CNs handle access requests from applications, decomposes tasks, and schedules task shards for parallel execution across DNs. GTMs manage and synchronize metadata for distributed transactions. DNs store business data and execute data queries. Among these, CNs and DNs perform the majority of the workload in a job and are therefore more prone to encounter anomalies, compared with GTMs. This

Authorized licensed use limited to: University of Electronic Science and Tech of China. Downloaded on October 13,2026 at 12:18:11 UTC from IEEE Xplore. Restrictions apply.

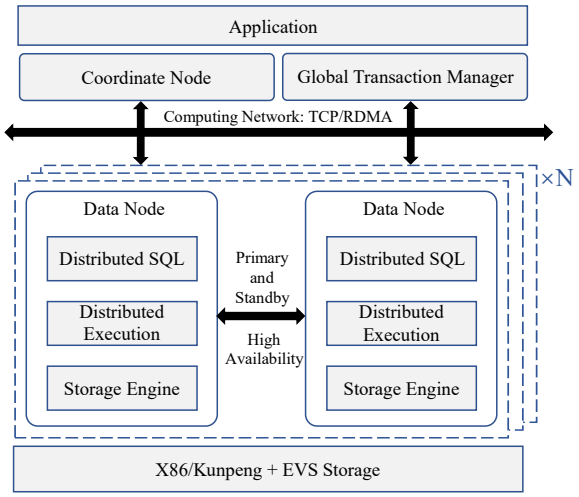


Fig. 1. The Distributed Architecture of GaussDB (DWS)

work focuses primarily on anomalies related to CNs and DNs. Due to the design of distribution, there are some anomalies different from those in single-machine databases. The skewed distribution of data tables across DNs leads to inefficient transaction execution and the overall performance decreases when there are too many connections in CNs that occupy too much resource.

To guarantee the service availability, GaussDB (DWS) is equipped with an efficient monitoring system that continuously tracks and collects KPIs and logs of databases, facilitating post-issue reviews. GaussDB (DWS) also incorporates rule-based and deep learning-based anomaly alert systems, enabling real-time detection of anomalies via predefined rules and deep model inference. Users can also report anomalies manually. However, the information provided by alerts typically does not directly reveal the root cause of the anomaly.

B. KPI and KPI Group

The monitoring system collects hundreds of system-level KPIs and database-level KPIs. Each KPI $K \in \mathcal{K}$ reflects a specific aspect of system performance, and collectively they indicate the overall state of the system. Experienced DBAs select 57 KPIs that are closely related to anomalies. Some KPIs describe the same aspect of the system from different perspectives. For instance, "CPU usr" and "CPU idle" both provide insights into the CPU component but from different perspectives. These KPIs exhibit similar behaviors and are likely to be correlated. To incorporate this prior knowledge and enhance the modeling of correlations between KPIs, we group KPIs based on the aspects of the system state they describe. For each group $G \in \mathcal{G}$, all $K \in G$ describe the same aspect of the system state. Tab. I lists the selected KPIs in group.

C. Problem Statement

The monitoring system collects time series data of KPIs at fixed intervals for each cluster. For a cluster c, the KPI data at time t is represented as $K_t^c = [k_t^1, k_t^2, \ldots, k_t^n]$, where n is the number of KPIs. The KPI window at time t is defined as the consecutive KPI data starting from t: $K_{t:t+\delta}^c = [K_t^c, K_{t+1}^c, \ldots, K_{t+\delta}^c]$, where δ is the window size. The set of root causes is defined as $Y = \{y_1, y_2, \ldots, y_m\}$, where m is the number of root causes and each element in Y represents a root cause defined by DBAs based on long-term experience. Historical anomaly cases consist of KPI window recorded during anomaly and their corresponding root causes.

Unlike traditional anomaly diagnosis settings [17], historical normal data of clusters should also be provided. It is defined as the historical time series data of KPIs $N_{t_1,t_2}^c = [K_{t_1}^c, K_{t_1+1}^c, \ldots, K_{t_2}^c]$ over a DBA-verified anomaly-free time interval $[t_1, t_2]$. The selected interval aims to cover diverse normal patterns of KPIs as much as possible.

The anomaly diagnosis task is described as: given historical anomaly cases and historical normal data of clusters, for an input KPI window $K_{t:t+\delta}^c$ during anomaly, we determine its root cause from Y. Additionally, we provide two ordered lists, *SL* and *RL*, as part of the anomaly interpretation. Higher-ranking KPIs in *SL* are more likely to be the symptom KPIs and higher-ranking KPIs in *RL* are more likely to be the root cause KPIs.

TABLE I
KPI GROUP AND KPIS IN EACH GROUP

	Group	KPIs
Operating System Level	CPU	inst cpu, usr, sys, iowait, idle
	Disk I/O	average peak iops, average spill size, tps, kb wrtn/s, wkb/s, util
	Disk Usage	data disk skew, max data percentage, avg data percentage, avg system percentage, avg log percentage
	Memory	inst mem, mem used, buffers, cached, process used memory, max process memory, dynamic used memory, shared used memory, cstore used memory, comm used memory
	Network	receive bytes, tcp resend rate
Database System Level	Execution	total sessions count, active queries, connections count
	Table	total skew size, total dirty page size
	SQL Stats	delta tup returned, delta tup fetched, delta tup inserted, delta tup updated, delta tup deleted, delta xact commit, delta xact rollback, delta deadlocks
	Log	primary dn xlog, primary dn cbm, primary dn rewind bak, primary dn sqltmp, standby dn xlog, standby dn cbm, standby dn rewind bak, standby dn sqltmp, secondary dn xlog, secondary dn cbm, secondary dn sqltmp, secondary dn rewind bak, cn xlog, cn cbm, cn rewind bak, cn sqltmp

III. FRAMEWORK

In this section, we provide a detailed introduction to our proposed method. First, we present an overview of the anomaly diagnosis system we designed for distributed cloud database systems. Following that, we describe the four major components of our method in detail: data processing, learning via SDN, online diagnosis and anomaly interpretation.

Authorized licensed use limited to: University of Electronic Science and Tech of China. Downloaded on October 13,2026 at 12:18:11 UTC from IEEE Xplore. Restrictions apply.

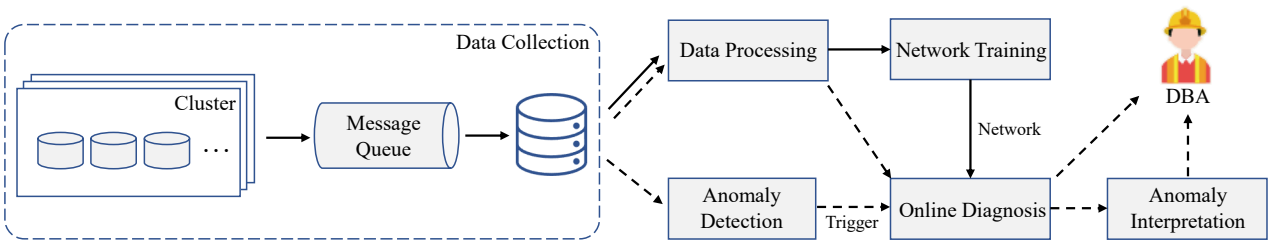


Fig. 2. Overall framework of the proposed method. The solid lines represent the offline process and the dashed lines represent the online process.

A. System Overview

Fig.2 illustrates the overview of our anomaly diagnosis system for distributed cloud database systems. In the data collection component, the monitoring system periodically collects KPIs data from each cluster and transfers it to the warehouse via a message queue for storage. During the offline training phase, historical anomaly cases and normal data are retrieved from the data warehouse and fed into the data processing component. In the data processing component, normal data of clusters is cleaned and segmented into KPI windows with the same length as anomaly KPI windows. Both anomaly and normal KPI windows are used to construct Hierarchical Dependency Graphs (HDGs) by correlation analysis. Each sample is composed of the KPI window and the corresponding HDG. These samples are then input into the Learning via SDN component (Network Training), where training samples are constructed and SDN is trained. The network learns high-level discrepancy representations between normal and anomaly samples, even with a small number of labeled anomaly samples. The trained network is subsequently used for online diagnosis. During the online phase, the system's anomaly detection component continuously monitors KPI data and detects anomaly in real time. Upon detecting an anomaly, the online diagnosis process is triggered. The KPI window of reported anomaly is taken from the data warehouse and processed through the data processing component. The trained SDN then diagnoses the anomaly. Finally, the anomaly interpretation component locates the symptom KPIs and the root cause KPIs. The diagnosis results with interpretations are presented to DBAs.

B. Data Processing

1) Data Cleaning: To robustly extract discrepancy between normal and anomaly samples, it is critical to ensure the purity of the normal samples, avoiding contamination by outliers that deviate from normal patterns. We adopt a classic and efficient outlier detection algorithm, Isolation Forest [23], to identify and label outliers in normal time series of KPIs of the input cluster. After labeling outliers, we remove them with their l adjacent points on both sides. The remaining time series is divided into multiple segments. We employ a sliding window approach to extract normal KPI windows with the length same as anomaly KPI windows. To minimize redundancy among the window, the step length of sliding is set to $\frac{\delta}{2}$.

2) Hierarchical Dependency Graph Construction: We observe that, during anomalies, the correlations between KPIs often differ significantly from those observed under normal conditions. These changes in correlations are influenced by the nature of the anomaly events. Cases of the same anomaly root cause tend to display similar correlation states, which are learnable patterns. A dependency graph is a graph-based data structure designed to represent correlation states, where nodes represent KPIs and edges represent correlations of KPIs. Different anomaly events often impact distinct system components. Therefore, modeling correlations at the KPI Group level can further enhance the network's ability to discriminate root cause. The process to construct a HDG is as follows:

KPI Dependency Graph. First, all KPIs are added to the graph as nodes. Subsequently, independences of all KPI pairs are tested, and edges are added between KPI nodes that exhibit dependency. Specifically, we employ the widely used Fisher-Z Test [24] for testing independences of KPIs. The Fisher-Z Test tests the independence based on the Pearson correlation coefficient by defining the test statistic Z:

$$Z = \frac{\sqrt{\delta - 3}}{2} \ln\left(\frac{1+r}{1-r}\right) \tag{1}$$

where r represents the Pearson correlation coefficient. If two KPIs are independent, Z follows a standard normal distribution. With this theory, the test evaluates the *p_value* of null hypothesis. If the *p_value* is less than 0.05, the two KPIs are considered to exhibit dependency.

Hierarchical Dependency Graph. After constructing the KPI dependency graph, we extend it to the HDG. This hierarchical graph consists of two layers: the lower layer represents the KPI dependency graph, and the upper layer represents the KPI Group dependency graph. In the KPI Group dependency graph, nodes correspond to KPI Groups, and edges derive from the KPI dependency graph. Specifically, for any two KPI Groups G_1 and G_2, an edge exists between them in the upper layer if there exists $K_1 \in G_1$ and $K_2 \in G_2$ such that an edge exists between K_1 and K_2 in the KPI dependency graph.

C. Design of Siamese Discrepancy Network

1) Overview: The Siamese network [25] is a well-known framework for similarity comparison in few-shot learning fields. It enables learning by comparing a small number of samples without requiring extensive labeled data. Its powerful

Authorized licensed use limited to: University of Electronic Science and Tech of China. Downloaded on October 13,2026 at 12:18:11 UTC from IEEE Xplore. Restrictions apply.

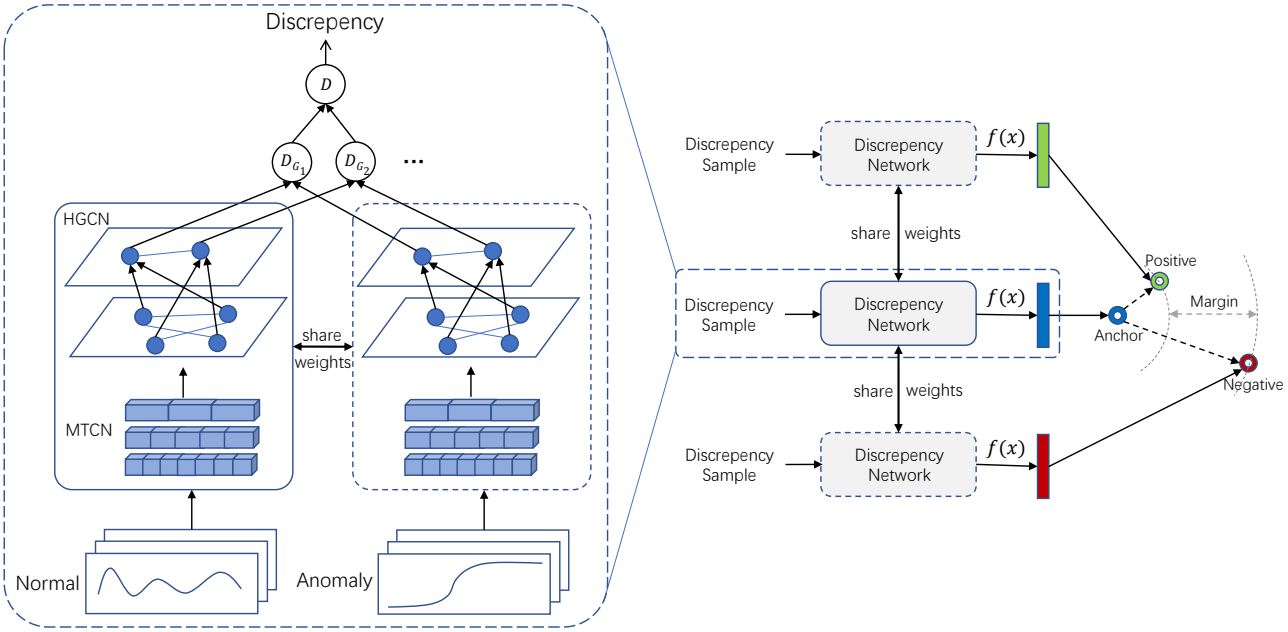


Fig. 3. The Architecture of SDN

representation learning capability allows it to extract high-level representations that exhibit both powerful separability and discriminability. Based on it, we design SDN tailored for the anomaly diagnosis in distributed cloud databases. SDN consists of three weight-sharing discrepancy networks, which learn high-level discrepancy representations between normal and anomaly samples. By employing the Triplet Loss [22], the network learns discrepancy representations in a specific Euclidean space, where representations with the same root cause are close, and those with different root causes are far. This enhances both the separability and discriminability of the network. Each discrepancy network is composed of two feature extraction components with shared weights and a discrepancy representation network. Two feature extraction components extract features from normal and anomaly samples respectively. The shared weights not only reduce the number of parameters, but more importantly ensure that the features are extracted from the same perspective for both normal and anomaly samples. The consistency is crucial for effective comparison. For example, to identify the anomalies with higher level, the level features of the anomaly sample and the normal sample should be extracted in the same way. Next we provide a detailed description of each component in the discrepancy network.

2) Feature Extraction: The feature extraction component is used to extract the features of the normal or anomaly sample. It consists of a MTCN and a HGCN.

Multi-scale trend CNN. CNNs [26] are widely used to extract features of images while reducing the number of parameters. Because time series and images both have sparse information density, 1-D CNN [27] is also commonly used to extract the features of time series, which is more efficient than the commonly used RNN while maintaining a competitive performance. However, there are notable differences between time series and images. For time series, trends and shapes are often of greater importance. In databases, anomalies in KPIs typically manifest as sudden spikes/drops and sustained increases/decreases. Additionally, consistently high or low level of KPIs may also indicate an anomaly. Based on these observations, we designed MTCN to explicitly extract both short-term and long-term trends as well as level features from time series. Specifically, for the input time series a trend CNN employs a sliding window to segment the series into windows of size θ each of which is denoted by $TW = [x_0, x_1, \ldots, x_\theta]$, similar to 1-D CNN. For each window, linear regression is applied to calculate the regression coefficient f and the intercept e.

$$f = \frac{\theta \sum_{t=0}^{\theta-1} (t x_t) - \sum_{t=0}^{\theta-1} t \sum_{t=0}^{\theta-1} x_t}{\theta \sum_{t=0}^{\theta-1} (t^2) - \left(\sum_{t=0}^{\theta-1} t\right)^2} = \frac{6 \sum_{t=0}^{\theta-1} (2t - \theta + 1) x_t}{\theta (\theta^2 - 1)} \quad (2)$$

$$e = x_0 \quad (3)$$

where f and e can be considered as a trend transformation and an identity transformation of the input time series.Subsequently, a 1-D convolution layer with a kernel size of θ is applied to f and e to extract further features.

$$Z_f = f * \mathbf{\Gamma}_{h \times \theta \times 1} \quad (4)$$

$$Z_e = e * \mathbf{\Gamma}_{h \times \theta \times 1} \quad (5)$$

Authorized licensed use limited to: University of Electronic Science and Tech of China. Downloaded on October 13,2026 at 12:18:11 UTC from IEEE Xplore. Restrictions apply.

where $\mathbf{\Gamma}_{h\times\theta\times 1}$ represents a 1-D convolution filter with h channels and a kernel size of $\theta \times 1$, and $*$ denotes the convolution operation.

In CNNs, pooling operations are commonly used to reduce the number of parameters and enhance the robustness of the extracted features. The two most commonly used pooling operations are max pooling and average pooling [28]. Average pooling is effective at reducing the impact of local noise and extracting smooth features, which makes it more suitable for regression tasks. Max pooling, on the other hand, excels at extracting the most significant features, which is useful for identifying anomalies in time series so it is more suitable for the anomaly diagnosis task. Therefore, a max pooling layer is applied to aggregate the features along the time dimension after convolution. The trend CNN with scale θ maps the input to $Z_\theta \in \mathbb{R}^{2\times h}$. Multiple trend CNNs with different scales are used to capture diverse features, with the scale parameter $\theta \in \{3, 5, 7\}$. In summary, MTCN maps the input to $Z \in \mathbb{R}^{3\times 2\times h}$.

$$Z_\theta = \max(Z_f) \oplus \max(Z_e) \tag{6}$$

$$Z_{mtcn} = Z_3 \oplus Z_5 \oplus Z_7 \tag{7}$$

where $\oplus$ denotes concatenation.

Hierarchical GCN. GDN [4] has achieved excellent performance, demonstrating the effectiveness of graph neural networks in time series anomaly detection. In GDN, a learnable graph structure is used to explicitly model the correlations between nodes. However, due to the limited number of samples in anomaly diagnosis, it is difficult to learn an accurate graph structure. Therefore, we use a pre-built HDG to explicitly leverage the correlations of KPIs, instead of the self-learned graph. Additionally, we extend the GCN [29] to a HGCN to incorporate multi-granularity correlation information of KPIs. HGCN consists of two levels of GCNs and takes the output of MTCN and HDG as input. First, a nonlinear transformation is applied to the output of MTCN. The lower-level GCN applies graph convolution operations on the KPI dependency graph to aggregate node features and $X_K^{(i)}$ represents the feature vector of KPI node K after i-layer graph convolution transformation.

$$X_K^{(0)} = \text{ReLU}(W_K^{(0)} Z_{mtcn} + b_K^{(0)}) \tag{8}$$

$$X_K^{(i)} = \sigma(A_K X_K^{(i-1)} W_K^{(i)} + b_K^{(i)}) \tag{9}$$

where $W_K^{(0)}$ and $b_K^{(0)}$ are the parameters in the nonlinear transformation, A_K is the adjacency matrix of the KPI dependency graph, $W_K^{(i)}$ and $b_K^{(i)}$ are weights and constants of the i-th layer where $1 \le i \le p$.

Next, the KPI node features are aggregated according to the group to generate the corresponding KPI Group features, i.e., the node features of each $G \in \mathcal{G}$ are aggregated from all $K \in G$. The upper-level GCN then applies graph convolution operations on the KPI Group dependency graph to aggregate KPI Group node features and generate $X_G^{(q)}$.

$$X_G^{(0)=\text{ReLU}(W_G^{(0)} X_K^{(p)} + b_G^{(0)})} \tag{10}$$

$$X_G^{(i)} = \sigma(A_G X_G^{(i-1)} W_G^{(i)} + b_G^{(i)}) \tag{11}$$

where $W_G^{(0)}$ and $b_G^{(0)}$ are the parameters of the aggregation transformation, A_G is the adjacency matrix of the KPI group dependency graph, $W_G^{(i)}$ and $b_G^{(i)}$ are weights and constants of the i-th layer where $1 \le i \le q$. The numbers of graph convolution layers p and q depend on the complexity of data.

3) Discrepancy Representation: For the KPI group features output by feature extraction components, we employ a KPI Group-independent feedforward layer to model discrepancy of KPI groups. The two sets of features corresponding to the same KPI group from feature extraction components are aggregated to generate the local discrepancy representation D_G of that KPI Group in the feedforward layer. The discrepancy representations of KPI Groups are non-overlapping within the feedforward layer. Finally, a multi-layer fully connected network is applied to summarize the discrepancy representations of all KPI Groups, producing the global high-level discrepancy representation D.

$$D_G = \text{ReLU}(W_{D_G}({}_{(Normal)}X_G^{(q)} \oplus_{(Anomaly)} X_G^{(q)}) + b_{D_G}) \tag{12}$$

$$D = \text{FCN}(\oplus_{G\in\mathcal{G}} D_G) \tag{13}$$

where ${}_{(Normal)}X_G^{(q)}$ and ${}_{(Anomaly)}X_G^{(q)}$ denotes the KPI Group node features extracted by two feature extraction components, and FCN denotes the fully connected network.

D. Learning via Siamese Discrepancy Network

In this section, we introduce the customized sample construction method and network optimization strategy designed to support the training of SDN.

1) Sample Construction: To learn discrepancy representations we need to pair anomaly samples with normal samples from the same cluster to form training samples which are also called discrepancy samples denoted as *DS*. The root cause of the anomaly sample serves as the label of the discrepancy sample.

$$DS = (Normal^c, Anomaly_a^c) \tag{14}$$

where a denotes root cause, and c denotes the cluster in which the sample is collected. To learn discrepancy representations invariant to normal patterns, we select k normal samples of the corresponding cluster for each anomaly sample. This approach fully leverages the prior correspondence between samples and clusters, and effectively expands the sample size by k times. Given a large number of normal samples within a cluster, we propose the following two downsampling strategies to construct discrepancy samples.

Cluster-based Strategy. To learn discrepancy representations invariant to normal patterns, we aim to preserve the diversity of normal sample patterns as much as possible during sampling. For each cluster, we employ K-medoids [30] on normal samples and then select sample cluster centers as representative samples. The rationale is that samples within the same sample cluster tend to share similar patterns, resulting in redundant information, whereas samples from different clusters are more likely to exhibit diverse patterns. K-medoids is

Authorized licensed use limited to: University of Electronic Science and Tech of China. Downloaded on October 13,2026 at 12:18:11 UTC from IEEE Xplore. Restrictions apply.

an improvement of K-means [31] which selects actual samples as cluster centers, making it more robust to noise and outliers, thereby ensuring the validity of the sampled normal samples. Additionally, K-medoids supports customizable distance metrics. We adopt channel-independent euclidean distance as the metric to measure the distance between normal samples which emphasizes the contribution of single-channel distance to the overall distance. The channel-independent euclidean distance is defined as:

$$d_{ci}(x,y) = \sum_{c=1}^{n} \sqrt{\sum_{t=1}^{m} (x_{t,c} - y_{t,c})^2} \tag{15}$$

where m denotes the length of window, and n denotes the number of KPIs. Finally, each anomaly sample is paired with representative normal samples of the corresponding cluster.

Random Strategy. For each anomaly sample, k normal samples are randomly selected without replacement from normal samples of the corresponding cluster. These selected normal samples are then paired with the anomaly sample to form discrepancy samples.

2) Network Optimization: Siamese networks are usually trained by minimizing Contrastive Loss [32] or Triplet Loss [22] to learn a representation with both powerful separability and discriminability. Triplet Loss is an improvement of Contrastive Loss, as it makes better use of the relative relationships among samples. By simultaneously considering an anchor, a positive sample, and a negative sample, Triplet Loss effectively learns finer-grained feature representations, capturing both the similarity and difference between samples. Triplet Loss helps to achieve finer discrimination of similar root causes.

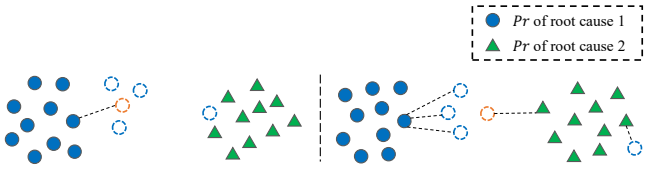
$$L = \frac{1}{N} \sum_{i}^{N} \left[d(f(x_i^a), f(x_i^p)) - d(f(x_i^a), f(x_i^n)) + \alpha \right]_+ \tag{16}$$

where x_i^a, x_i^p and x_i^n denote the anchor, the positive sample and the negative sample respectively, and $d(\cdot,\cdot)$ denotes the euclidean distance.

Specially, a triplet consists of a discrepancy sample (anchor), a discrepancy sample with the same root cause (positive sample), and a discrepancy sample with a different root cause (negative sample). The number of combinations is very large, leading to an exponential growth in the training data. However, due to the sharing of weights, the number of parameters does not increase significantly, effectively alleviating overfitting.

Traversing all triplets is highly inefficient during training. Therefore, to accelerate network convergence, we adopt the Online Triplet Generation strategy proposed in FaceNet [22]. Specifically, in each mini-batch, for each discrepancy sample x_i^a, we randomly select a same-type discrepancy sample as the positive sample x_i^p and select its hard negative sample $argmin_{x_i^n} d(x_i^a, x_i^n)$ for gradient descent.

With Triplet Loss, SDN learns the discrepancy representations in a specific euclidean space, where the discrepancy representations with the same root cause are close, and those with different root causes are far. Intuitively, discrepancies between anomalies with the same root cause and normal samples are similar, whereas discrepancies between anomalies with different root causes and normal samples are distinguishable.



(a) A example of prototype-based strategy (b) A example of election-based strategy

Fig. 4. Two examples of strategies for determining the root cause. The $k(k = 5)$ dashed objects represent discrepancy representations of the testing sample, the dashed line represents the nearest neighbor relationship and the orange object represents the PR of the testing sample in each subfigure.

E. Online Diagnosis

To diagnose the testing sample, we need to identify the training samples that have similar discrepancy representations with the test sample, as they are most likely to have the same root cause. We apply the sample construction method to pair the testing sample with normal samples of the corresponding cluster, resulting in a set of discrepancy samples. To ensure the consistency of inference results, we consider only the cluster-based method for construction. The constructed k discrepancy samples are input into the discrepancy network to generate k discrepancy representations.

Since there are k discrepancy representations for each anomaly sample, to improve efficiency and mitigate the impact of outliers, we define the representation with the smallest total distance to the other representations as the prototype discrepancy representation of the anomaly sample, denoted as *PR*.

$$PR = \arg\min_{D_i \in D} \sum_{D_j \in D} d(D_i, D_j) \tag{17}$$

where D represents a set of discrepancy representations.

Given k discrepancy representations of the testing sample and *PR*s of training samples, we propose two strategies for determining the root cause with less impact of outliers:

Prototype-based Strategy. Compute the *PR* of the testing sample. The sample in training set whose *PR* is nearest to that of the testing sample is then selected. If the distance is less than the threshold τ, the testing sample is diagnosed as the root cause same as the selected sample, otherwise as a unknown root cause. An example is depicted in Fig. 4(a). In it, one discrepancy representation of the testing sample which is an outlier is nearest to *PR*s of training samples with root cause 2 and others are near to *PR*s of training samples with root cause 1. In this case, the correct diagnosis result is guaranteed by selecting the *PR* of the testing sample.

Election-based Strategy. For each discrepancy representation of the testing sample, the prototype discrepancy representation of the nearest training sample is identified. Then a

Authorized licensed use limited to: University of Electronic Science and Tech of China. Downloaded on October 13,2026 at 12:18:11 UTC from IEEE Xplore. Restrictions apply.

majority vote is conducted based on the labels of these nearest neighbors. If more than half of the nearest neighbors label the same root cause as that of the testing sample, the testing sample is diagnosed as that root cause, otherwise as a unknown root cause. The rationale is that the discrepancy representations of a sample with learned root cause are consistent to various normal patterns. Specifically, the discrepancy representations between a anomaly sample with seen/unseen root cause and different normal samples tend to be clustered/scattered in the feature space. An example is depicted in Fig.4(b). In it, the *PR* of the testing sample is nearest to *PR*s of training samples with root cause 2, but the majority of discrepancy representations of the testing sample is near to *PR*s of training samples with root cause 1. In this case, the correct diagnosis result is guaranteed by a majority vote.

F. Anomaly Interpretation

Benefiting from learning of discrepancy representations, SDN has the capability to locate the root cause KPIs of anomalies. To locate the root cause KPIs, it is necessary to locate the symptom KPIs first, as the root cause KPIs usually lie within the set of symptom KPIs. The locating process proceeds as follows: 1) given the prototype discrepancy sample *DS* of the testing sample, for each $K \in \mathcal{KPI}$, replace the time series of K in the anomaly sample with that in normal sample, and the result is denoted as $\widehat{DS}$. 2) Input $\widehat{DS}$ into discrepancy network to get modified discrepancy representations. Then, compute the deviation *Dev* between the modified and original representations as:

$$Dev = d(f(DS), f(\widehat{DS})) \tag{18}$$

where the greater the deviation of $\widehat{DS}$ replaced with K is, the more anomalous the K is. *Dev* can be considered as anomaly scores of KPIs. The rationale is that for normal KPIs in the anomaly sample, their patterns are similar to those of normal samples, so replacing them does not significantly affect the discrepancy representations. In contrast, replacing an anomalous KPI with a normal KPI substantially affect the discrepancy representation. By sorting all KPIs in ascending order of *Dev*, we obtain a ranked anomaly list *SL*. The higher ranked KPIs in *SL* is more likely to be symptom KPIs. Typically, the top few KPIs in *SL* are considered as the symptom KPIs.

To further pinpoint the root cause KPIs, a two-step approach is commonly used where the first step involves constructing a causal graph by causal inference techniques such as the PC algorithm [33] and the second step uses the constructed causal graph to generate a global importance ranking of nodes by the PageRank algorithm [34]. However, due to the challenge that the presence of latent variables make it difficult to accurate causal inference, we adopt the Weighted Undirected Dependency Graph (WUDG) proposed in FluxInfer [12] instead of causal graph. We extend the KPI dependency graph obtained during hierarchical dependency graph construction into a WUDG by completing missing edges between nodes and assigning the reciprocal of *p_value* from Fisher-Z Test as the weights of edges. Then, we adopt the Personalized PageRank algorithm [35] instead of the standard PageRank algorithm treating each node equally to better capture the propagation of anomalies. Personalized PageRank allows preference specification for certain nodes, making it more suitable to anomaly propagation scenarios. In this approach, we compute the Personalized PageRank Vector (*PPV*) as the root cause score for each node. To compute the *PPV*, we define the transition probability between nodes as $P_{ij} = \frac{\omega_{ij}}{\sum_j \omega_{ij}}$ where ω_{ij} denotes the weight of the edge between nodes i and j. Then we calculate the normalized anomaly scores of KPIs as the node preference vector π, where $\pi_i = \frac{Dev^{(i)}}{\sum_i Dev^{(i)}}$, so that KPIs with large anomaly scores are more likely to be visited during random transitions. Formally, *PPV* on WUDG is computed as:

$$PPV(v) = (1-\alpha) \cdot \sum_{u \in N(v)} P_{uv} \cdot PPV(u) + \alpha \cdot \pi \tag{19}$$

where α is the damping factor usually set to 0.15, and $N(v)$ denotes the set of nodes directly connected to v. Finally, KPIs in *RL* are in ascending order of the calculated *PPV*. KPIs ranked higher in *RL* are more likely to be root cause KPIs.

TABLE II
THE ROOT CAUSES WITH THE NUMBERS OF CASES IN REAL WORLD AND DBPA-MINI DATASETS

Dataset	No.	Root cause	#Cases
Real World	1	Distributed Table Skew	22
	2	Distributed Computation Skew	21
	3	I/O-intensive Task Overload	25
	4	Single Task CPU Overload	22
	5	Single Task Memory Overload	24
	6	Highly Concurrent Task CPU Overload	24
	7	Highly Concurrent Task Memory Overload	20
	8	CN Connection Overload	23
	9	Highly Concurrent Poorly Written Query	27
		Total	208
DBPA-Mini	1	Highly Concurrent Inserts	24
	2	Missing Indexes	24
	3	Heavy Workloads	24
	4	Vacuum	24
	5	Highly Concurrent Commits	24
	6	Lock Waits	24
	7	Too Many Indexes	24
	8	Small Shared Buffer	24
	9	I/O Saturation	24
		Total	216

IV. EVALUATION

A. Experiment Settings

SDN is implemented in Python3.10 with PyTorch 2.0.1. All experiments are run on a Linux (Ubuntu 18.04LTS) machine with a NVIDIA Tesla V100 GPU card. We use Adam to train SDN and set the initial learning rate to 0.01 and weight decay to 0.01. To avoid gradient exploding, we clip the gradients of all parameters at 1. We train each model for 20 epochs and

Authorized licensed use limited to: University of Electronic Science and Tech of China. Downloaded on October 13,2026 at 12:18:11 UTC from IEEE Xplore. Restrictions apply.

TABLE III
ANOMALY DIAGNOSIS RESULTS

Method			DBPA-Mini			Real World		
			Macro Pre(%)	Macro Rec(%)	Macro F1(%)	Macro Pre(%)	Macro Rec(%)	Macro F1(%)
Statistics-based	DBSherlock		48.41	38.89	34.18	51.10	44.21	40.32
	iSQUAD		45.62	44.44	44.62	58.30	49.42	48.66
Deep Learning-based	MLSTM-FCN		82.23	78.15	79.27	78.37	72.64	74.59
	InceptionTime		84.67	80.59	81.35	80.85	75.89	77.33
	SANet		79.41	74.87	75.87	74.23	69.38	70.85
	SDN	R-P	87.02	86.67	86.73	83.01	82.78	82.76
		R-E	88.58	88.33	88.32	85.59	85.56	85.52
		C-P	89.15	89.00	88.91	87.40	87.22	86.89
		C-E	**89.54**	**89.45**	**89.39**	**87.93**	**87.78**	**87.67**

set the batch size to 32. We use a publicly available dataset and a real world dataset from Huawei to evaluate our model. A profile of the dataset is given in the Tab. II.

- **DBPA-Mini:** The dataset is a part of the public dataset DBPA [36]. DBPA is a synthetic dataset for single-machine OLTP database anomaly diagnosis. It contains 9 kinds of root causes related to different impact factors, and collects a total of 110 system level and database level KPIs. It is built by reproducing anomalies in the testing environment. DBPA has a large number of samples. Due to the lack of anomaly samples in the real world, we only extract 2% samples from DBPA to get DBPA-Mini.
- **Real World:** The dataset is constructed by simulating real anomaly cases on Huawei Cloud GaussDB (DWS). We create three clusters simulating clusters of different real-world businesses. Then we reproduce the root causes of historical anomaly cases and collect data. With long-term experience of DBAs, the root causes are classified into nine most common types. The dataset uses 57 KPIs related to anomaly, which are carefully selected by the DBA from hundreds of KPIs. The sampling interval of KPI data is 30 seconds, which is enough to show the diverse changes of KPIs.

The training set and testing set of the two datasets are divided with the ratio of $5:5$. Due to the small size of dataset, in order to make full use of the data, we used 5-fold cross validation for training. The testing set is evaluated after the best model parameters are obtained by validation. We repeat each experiment three times due to the stochastic training and present the average results.

Baselines: To study the diagnosis performance of SDN, we compare it with the following methods:

- **DBSherlock:** [14] It is a performance diagnosis framework for single-machine databases, which combines predicate mining and causality analysis technology. By comparing values of normal and anomalous region, it mines heuristic rules and constructs causal models for anomaly diagnosis.
- **iSQUAD:** [20] It is an anomaly diagnosis method for slow queries in cloud databases. It extracts several predefined anomalous patterns of KPIs by statistical method according to which samples discretized. Then clusters the historical samples to label the root cause, and diagnosis the testing sample as the root cause of the nearest cluster during diagnosis.
- **MLSTM-FCN:** [37] It is a deep learning model for multivariate time series classification combining long short-term memory network and fully convolutional network. The fully convolutional network is enhanced by the long short-term memory recurrent neural network sub-module, and the fully convolutional block is increased by using the squeeze excitation block.
- **InceptionTime:** [38] It is a deep convolutional network ensemble model for multivariate time series classification. Based on Inception network, it captures features on different time scales through parallel convolution kernels of different sizes.
- **SANet:** It is a deep network based on self-attention mechanism proposed in Transformer [39] and widely used in time series analysis. It captures local and global features efficiently and flexibly through self-attention mechanism. In our experiments, it consists of 2 encoder layers and a fully connected output layer. Each encoder layer consists of 4 attention heads.

Evaluation Metrics: We use Macro average of Precision (Macro Pre), Recall (Macro Rec) and F1-score (Macro F1) to evaluate the performance of the method.

In particular, in order to study the open set recognition capability of SDN, we use the True Accept Rate (*TAR*) and True Reject Rate (*TRR*) metrics which are commonly used in the field of face recognition. *TAR* refers to the proportion of testing samples correctly diagnosed the root causes of which exist in the training set. *TRR* refers to the proportion of testing samples correctly rejected whose root causes are absent in the training set.

$$TAR = \frac{|X_j \in X_g, f(X_j) = y_j|}{|X_g|} \tag{20}$$

$$TRR = \frac{|X_j \in X_n, f(X_j) = \textit{reject}|}{|X_g|} \tag{21}$$

where $j, g \in Y$, $n \notin Y$ and $f(\cdot)$ denotes diagnosis result.

Authorized licensed use limited to: University of Electronic Science and Tech of China. Downloaded on October 13,2026 at 12:18:11 UTC from IEEE Xplore. Restrictions apply.

TABLE IV
ANOMALY DIAGNOSIS RESULTS UNDER THE GENERALIZATION SETTINGS

Method	Cluster1-Others			Cluster2-Others			Cluster3-Others			ALL		
	Macro Pre(%)	Macro Rec(%)	Macro F1(%)	Macro Pre(%)	Macro Rec(%)	Macro F1(%)	Macro Pre(%)	Macro Rec(%)	Macro F1(%)	Macro Pre(%)	Macro Rec(%)	Macro F1(%)
DBSherlock	8.06	22.22	11.16	19.19	28.89	19.86	13.61	32.22	18.61	51.10	44.21	40.32
iSQUAD	35.76	37.78	36.01	46.01	43.33	43.42	44.14	43.33	42.66	58.30	49.42	48.66
MLSTM-FCN	58.81	62.93	59.68	59.63	64.42	60.94	57.69	62.15	59.89	78.37	72.64	74.59
InceptionTime	56.21	60.79	57.62	58.72	63.35	58.87	57.03	61.89	58.41	80.85	75.89	77.33
SANet	55.74	59.32	55.61	58.19	62.04	58.43	57.36	60.87	57.12	74.23	69.38	70.85
SN	71.18	72.67	70.96	71.54	72.22	71.55	70.38	72.11	71.75	84.58	84.21	83.73
SDN	**80.65**	**83.89**	**79.84**	**79.12**	**85.11**	**81.06**	**82.01**	**82.33**	**80.96**	**87.93**	**87.78**	**87.67**

We adopt the *HitRate@P%* metric proposed in OmniAnomaly [3] to study the performance of anomaly interpretation.

$$HitRate@P\% = \frac{Hit@\ \lfloor P\% \times |GT_t| \rfloor}{|GT_t|} \tag{22}$$

Hit@P% equals the number of overlapping dimensions between GT_t (ground truth) and the top $\lfloor P\% \times |GT_t| \rfloor$ contributing dimensions in *SL* or *RL* suggested by SDN.

Parameter Settings. For both DBPA-Mini and Real World, the k is set to 5 for discrepancy sample construction. In MTCN, the number of convolution kernel channels is set to 4 for DBPA-Mini and 6 for Real World. The number of graph convolution layers p and q in the HGCN component are both set to 2 for both two datasets. The dimension of the discrepancy representation D in the difference representation component is set to 64 for DBPA-Mini and 32 for the Real World dataset. The margin of the Triplet Loss is set to 1.0 during training for both datasets. The threshold τ in the online diagnosis is determined by grid search.

B. Overall Performance

In Tab. III, we present the results of anomaly diagnosis. As described in Sec.III, SDN has two sample construction strategies and two diagnosis strategies. They form four combinations, denoted as R-P, R-E, C-P, and C-E, where R denotes the random strategy, C denotes the cluster-based strategy, P denotes the prototype-based strategy, and E denotes the election-based strategy.

We can see that DBSherlock and iSQUAD perform much worse than the others. The main reason is that they extract a limited number of predefined anomaly features through statistical methods, which is difficult to deal with complex anomaly scenarios. In complex anomaly scenarios, KPIs change diversely, and the predefined anomaly features cannot cover all possible anomaly features. Specifically, DBSherlock extracts a fixed form of value anomaly features, but under multi-configuration datasets, the values of KPIs usually change in different ranges, and it is difficult to extract a unified value anomaly features for the same root cause. iSQUAD extracts four kinds of predefined trend anomaly features. Due to the diversity of trends in KPIs, diverse anomaly features cannot be effectively and fully extracted. In addition, trend anomaly features generalize better than value anomaly features in database anomalies, so iSQUAD performs better than DBSherlock.

Deep learning-based methods perform better than statistics-based methods, but worse than SDN. Deep learning models have strong learning capability, which allows adaptively learning rich anomaly patterns from data without much prior knowledge, so they are superior to statistics-based methods. However, due to the small data scale in anomaly diagnosis scenarios, deep learning models face overfitting issues. SANet models the global dependence of time series of KPIs through the self-attention mechanism, in which the multi-head attention module and the feedforward layer depend on a large number of parameters. It requires large amounts of data, so its performance is suboptimal. InceptionTime uses multi-scale convolution to extract local and global features of time series of KPIs. In InceptionTime, the bottleneck layer is used before the convolution operation and the Inception module is reused in the depth direction of the network. These designs reduce the complexity of the model and have a lower tendency of overfitting than SANet and MLSTM-FCN, so InceptionTime performs relatively better.

SDN achieves the best performance on two datasets. SDN substantially expands the training samples by the construction of discrepancy samples and the triplet optimization objective. It greatly reduces the tendency of the model to overfit. Moreover, the customized MTCN and HGCN utilize the prior knowledge to improve the performance of feature extraction. Specifically, among the four strategies of SDN, the cluster-based strategy outperforms the random strategy. The election-based strategy is slightly better than the prototype-based strategy, mainly because the election-based strategy can avoid the influence of extreme samples in some cases. C-E strategy is used in the subsequent experiments for SDN.

In the experiments conducted on the Real World dataset, we also record the training time and average diagnosis time for both the proposed method and the baselines. The results are shown in Tab. V. While the proposed method is not the most efficient in terms of training time, its performance is still acceptable. Additionally, the proposed method achieves diagnosis times on the order of milliseconds. Unlike the baselines, the proposed method requires inference on multiple

Authorized licensed use limited to: University of Electronic Science and Tech of China. Downloaded on October 13,2026 at 12:18:11 UTC from IEEE Xplore. Restrictions apply.

TABLE V
EVALUATION RESULTS OF TRAINING TIME AND DIAGNOSIS TIME

Method	Training Time(s)	Diagnosis Time(s)
DBSherlock	306.8	10.57
iSQUAD	8.9	0.92
MLSTM-FCN	868.0	0.42
InceptionTime	1078.1	0.45
SANet	431.5	0.16
SDN	759.3	0.79

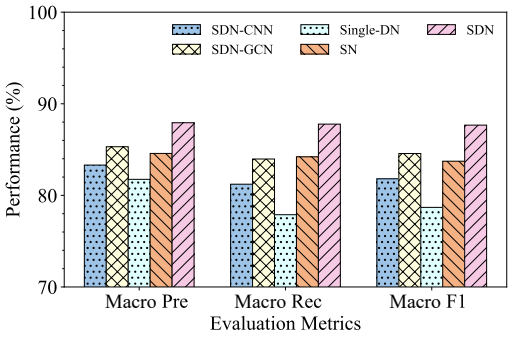


Fig. 5. Anomaly Diagnosis Results of Variants of SDN

discrepancy samples when diagnosing an anomaly. However, these can be processed in batches, ensuring that the diagnosis time meets real-time requirements.

C. Ablation Study

To evaluate the effects of different components in our model, we compare different variants of SDN on the Real World dataset, including:

- **SDN-CNN**: The MTCN in SDN is replaced by a general 1-D CNN.
- **SDN-GCN**: The hierarchical GCN in SDN is replaced by a general GCN. Eliminate the impact of KPI groups.
- **Single-DN**: Only a single DN from the siamese DN is kept and trained by cross-entropy loss.
- **SN**: Only one of the feature extraction components is kept and only the anomaly samples are taken as input.

The results are shown in Fig.5. Compared with the general CNN, the multi-scale trend CNN in SDN can extract the multi-scale features of time series and explicitly model the trend. Therefore, the Macro F1 of SDN is 7.2% higher than that of SDN-CNN. Compared with the general GCN, the hierarchical GCN of SDN introduces the prior knowledge related to KPIs, which enhances the modeling capability of correlation of KPIs. Therefore, the Macro F1 of SDN is 3.7% higher than that of SDN-GCN. Single-DN has poor performance. This is mainly because the training sample amount dropped substantially after removing the Siamese design, which leads to overfitting of the model. Compared with SDN, SN does not have the capability to learn discrepancy so does not perform well when KPIs have relative anomaly manifestation. Therefore, the Macro F1 of SDN is 4.7% higher than that of SN.

D. Generalization Study

To verify the generalization of our model, we conducted generalization experiments on the Real World dataset. Specifically, we choose two of the clusters for training and one for testing, and there are three settings.

Tab. IV shows the results. A-B denotes the setting of training on B and testing on A. ALL denotes the setting of training and testing on all clusters. DBSherlock has very poor generalization performance. Because it is designed for single-machine databases. It extracts rules by mining the value features of KPIs. However, the KPI values of clusters with different configurations vary in different ranges, which makes it impossible to extract effective rules. It is not suitable for multi-cluster scenarios. Compared with the setting of ALL, iSQUAD does not perform much worse under the generalization settings. This is mainly because iSQUAD extracts trend features of KPIs. In most anomaly scenarios, the trends of some KPIs are similar across clusters. For the three deep learning-based baselines, the performance decreases substantially under the generalization settings, with the Macro F1 at most 25.5% lower than that under the ALL setting. Because they directly learn discriminative features between different root causes without considering the differences of clusters. The learned features generalize poorly. To eliminate the effect of overfitting, we conducted experiments on SN. The performance under generalization settings is still substantially worse when using the triplet loss to avoid model overfitting with the Macro F1 at most 15.3% lower than that under the ALL setting. SDN has the best generalization performance. It learns discrepancy features with respect to clusters. Under the generalization settings the Macro F1 of SDN is at most only 8.9% lower than that under the ALL setting.

E. Open Set Study

In the real world, the training set cannot cover all root causes. When a new root cause appears, DBA hopes the model can correctly reject it. This problem is defined as open set recognition and is commonly seen in the field of face recognition [40]. We conducted experiments on the Real World dataset to verify that our model has the capability of open set recognition. Specifically, under the open set setting, we remove one root cause from the training set and conduct experiments and there are 9 settings in total. Since the prototype-based strategy and the election-based strategy have different strategies for rejecting unknown samples, we conducted experiments with SDN of both strategies.

The results are shown in Tab. VI. In most of the open set settings, SDN with the election-based strategy exhibits higher TAR but lower TRR than that of SDN with the prototype-based strategy. This indicates that SDN with election-based strategy has better capability to identify close set but has poor ability to reject unknown samples. In a comprehensive perspective, the Macro F1 of SDN with election-based strategy under the open set settings is still close to that under the closed set setting.

Authorized licensed use limited to: University of Electronic Science and Tech of China. Downloaded on October 13,2026 at 12:18:11 UTC from IEEE Xplore. Restrictions apply.

TABLE VI
ANOMALY DIAGNOSIS RESULTS UNDER THE OPEN SET SETTINGS

Setting	Prototype-based			Election-based		
	TAR(%)	TRR(%)	Macro F1(%)	TAR(%)	TRR(%)	Macro F1(%)
w/o cause1	83.13	87.50	83.87	87.50	70.00	85.35
w/o cause2	82.35	82.50	82.38	85.38	75.00	84.21
w/o cause3	86.25	100.0	87.54	88.13	97.50	88.70
w/o cause4	82.15	92.50	82.22	84.40	82.50	83.42
w/o cause5	83.50	85.00	83.81	86.75	80.00	85.37
w/o cause6	85.13	77.50	84.62	84.50	72.50	83.81
w/o cause7	82.59	87.50	83.65	86.50	77.50	85.24
w/o cause8	85.31	100.0	86.93	86.25	100.0	87.78
w/o cause9	84.68	90.00	85.22	87.45	87.50	87.46

TABLE VII
ANOMALY INTERPRETATION RESULTS

	HitRate@100%	**HitRate@150%**
Symptom KPIs	70.56	91.11
Root Cause KPIs	75.74	86.48

F. Results of Anomaly Interpretation

To verify the performance of the anomaly interpretation module in our method, we conducted experiments on the Real World dataset. To obtain ground truth, we invited experienced DBAs to label symptom KPIs and root cause KPIs for each anomaly case in the dataset.

Tab. VII shows the results. For both the symptom KPIs and the root cause KPIs, the method has a high $HitRate@150\%$. The $HitRate@150\%$ is much higher than $HitRate@100\%$, which indicates that in many cases the symptom KPIs or root cause KPIs are not at the top of SL or RL, but not far from the top. In summary, the anomaly interpretation module can effectively provide useful information for DBAs.

V. RELATED WORK

The target of anomaly diagnosis is to identify the root cause, which varies in different research works. Some works are devoted to locating root cause KPIs while others identifying root cause types. Therefore, we categorize existing methods into two types based on their target: anomaly diagnosis methods oriented to root cause KPIs and anomaly diagnosis methods oriented to root cause types.

Root Cause KPIs-oriented: TcpRT [6] is a framework for real-time anomaly detection and anomaly diagnosis in Alibaba Cloud database, which uses the historical KPI data of each database instance to build an adaptive Cauchy distribution statistical model. PerfAugur [10] is a method for mining anomaly predicates in the cloud platforms. It proposes a robust statistics based anomaly mining method, and interprets anomalies through the mined predicates. TcpRT and PerfAugur focus solely on the level anomaly features of KPIs, without considering the correlations between them. ExplainIt [11], FluxInfer [12] and CauseRank [9] adopt causal analysis techniques to capture the propagation of anomalies through the correlations of KPIs. BALANCE [7] introduce the Explainable AI (XAI) technique into anomaly diagnosis and proposed a Bayesian linear attribution model for root cause localization, which is a novel contribution.

Root Cause Type-oriented: DBSherlock [14] and iSQUAD [20] are statistics-based anomaly diagnosis methods designed for database systems. DBsherlock captures level anomaly features of KPIs by mining predicates. iSQUAD extracts four predefined trend patterns by statistical testing. The features they extract are limited and fail to account for diverse anomaly patterns. Dundjerski et al. [15] proposed a rule-based end-to-end automatic anomaly diagnosis system for Azure SQL cloud database systems. All rules are manually designed for a specific database system, which requires significant manual effort and is difficult to transfer to other database systems. DBPA [36] proposes a method for anomaly diagnosis benchmark of database systems. It constructs datasets by reproducing anomalies in the test environment and collecting sufficient data, and alleviates the issue of lack of labels to some extent.

VI. CONCLUSION

We propose a novel approach for anomaly diagnosis in distributed cloud databases. Specially, we design a novel deep network SDN, to model the discrepancy. It takes both anomaly and normal data as input and we design MTCN and HGCN to enhance feature extraction capability based on the characteristics of anomalies in database systems. Furthermore, we propose a discrepancy-based diagnosis paradigm. The paradigm enables training sample construction, representation learning of discrepancy and online diagnosis. In addition, we develop an anomaly interpretation method based on SDN which effectively locates symptom KPIs and root cause KPIs. Extensive experiments demonstrate the superior performance of our method in anomaly diagnosis and anomaly interpretation.

ACKNOWLEDGEMENTS

This work was supported by NSFC Grant No. 62372194.

Authorized licensed use limited to: University of Electronic Science and Tech of China. Downloaded on October 13,2026 at 12:18:11 UTC from IEEE Xplore. Restrictions apply.

REFERENCES

[1] G. Zhang, C. Li, K. Zhou, L. Liu, C. Zhang, W. Chen, H. Fang, B. Cheng, J. Yang, and J. Xing, "Dbcatcher: A cloud database online anomaly detection system based on indicator correlation," in *ICDE*, 2023, pp. 1126–1139.

[2] M. Ma, S. Zhang, J. Chen, J. Xu, H. Li, Y. Lin, X. Nie, B. Zhou, Y. Wang, and D. Pei, "Jump-starting multivariate time series anomaly detection for online service systems," in *USENIX ATC*, 2021, pp. 413–426.

[3] Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun, and D. Pei, "Robust anomaly detection for multivariate time series through stochastic recurrent neural network," in *KDD*, 2019, pp. 2828–2837.

[4] A. Deng and B. Hooi, "Graph neural network-based anomaly detection in multivariate time series," in *AAAI*, 2021, pp. 4027–4035.

[5] B. Zheng, L. Ming, K. Zeng, M. Zhou, X. Zhang, T. Ye, B. Yang, X. Zhou, and C. S. Jensen, " Adversarial Graph Neural Network for Multivariate Time Series Anomaly Detection ," *TKDE*, vol. 36, no. 12, pp. 7612–7626, 2024.

[6] W. Cao, Y. Gao, B. Lin, X. Feng, Y. Xie, X. Lou, and P. Wang, "Tcprt: Instrument and diagnostic analysis system for service quality of cloud databases at massive scale in real-time," in *SIGMOD*, 2018, pp. 615–627.

[7] C. Chen, H. Yu, Z. Lei, J. Li, S. Ren, T. Zhang, S. Hu, J. Wang, and W. Shi, "BALANCE: bayesian linear attribution for root cause localization," *SIGMOD*, vol. 1, no. 1, pp. 95:1–95:26, 2023.

[8] X. Han, L. Zhang, Y. Wu, and S. Yuan, "On root cause localization and anomaly mitigation through causal inference," in *CIKM*, 2023, pp. 699–708.

[9] X. Lu, Z. Xie, Z. Li, M. Li, X. Nie, N. Zhao, Q. Yu, S. Zhang, K. Sui, L. Zhu, and D. Pei, "Generic and robust performance diagnosis via causal inference for OLTP database systems," in *CCGRID*, 2022, pp. 655–664.

[10] S. Roy, A. C. König, I. Dvorkin, and M. Kumar, "Perfaugur: Robust diagnostics for performance anomalies in cloud services," in *ICDE*, 2015, pp. 1167–1178.

[11] V. Jeyakumar, O. Madani, A. Parandeh, A. Kulshreshtha, W. Zeng, and N. Yadav, "Explainit! - A declarative root-cause analysis engine for time series data," in *SIGMOD*, 2019, pp. 333–348.

[12] P. Liu, S. Zhang, Y. Sun, Y. Meng, J. Yang, and D. Pei, "Fluxinfer: Automatic diagnosis of performance anomaly for online database system," in *IPCCC*, 2020, pp. 1–8.

[13] X. Liu, Z. Yin, C. Zhao, C. Ge, L. Chen, Y. Gao, D. Li, Z. Wang, G. Liang, J. Tan, and F. Li, "Pinsql: Pinpoint root cause sqls to resolve performance issues in cloud databases," in *ICDE*, 2022, pp. 2549–2561.

[14] D. Y. Yoon, N. Niu, and B. Mozafari, "Dbsherlock: A performance diagnostic tool for transactional databases," in *SIGMOD*, 2016, pp. 1599–1614.

[15] D. Dundjerski and M. Tomasevic, "Automatic database troubleshooting of azure SQL databases," *IEEE Trans. Cloud Comput.*, vol. 10, no. 3, pp. 1604–1619, 2022.

[16] K. Dias, M. Ramacher, U. Shaft, V. Venkataramani, and G. Wood, "Automatic performance diagnosis and tuning in oracle," in *CIDR*, 2005, pp. 84–94.

[17] B. Aksar, E. Sencan, B. Schwaller, O. Aaziz, V. J. Leung, J. M. Brandt, B. Kulis, M. Egele, and A. K. Coskun, "Runtime performance anomaly diagnosis in production HPC systems using active learning," *TPDS*, vol. 35, no. 4, pp. 693–706, 2024.

[18] Z. Li, N. Zhao, M. Li, X. Lu, L. Wang, D. Chang, X. Nie, L. Cao, W. Zhang, K. Sui, Y. Wang, X. Du, G. Duan, and D. Pei, "Actionable and interpretable fault localization for recurring failures in online service systems," in *ESEC/SIGSOFT FSE*, 2022, pp. 996–1008.

[19] Y. Zhang, Z. Guan, H. Qian, L. Xu, H. Liu, Q. Wen, L. Sun, J. Jiang, L. Fan, and M. Ke, "Cloudrca: A root cause analysis framework for cloud computing platforms," in *CIKM*, 2021, pp. 4373–4382.

[20] M. Ma, Z. Yin, S. Zhang, S. Wang, C. Zheng, X. Jiang, H. Hu, C. Luo, Y. Li, N. Qiu, F. Li, C. Chen, and D. Pei, "Diagnosing root causes of intermittent slow queries in large-scale cloud databases," *PVLDB*, vol. 13, no. 8, pp. 1176–1189, 2020.

[21] H. Cloud, "Huawei cloud product of gaussdb (dws)." [Online]. Available: https://www.huaweicloud.com/product/dws.html

[22] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *CVPR*, 2015, pp. 815–823.

[23] F. T. Liu, K. M. Ting, and Z. Zhou, "Isolation forest," in *ICDM*, 2008, pp. 413–422.

[24] X.-L. Meng, R. Rosenthal, and D. B. Rubin, "Comparing correlated correlation coefficients." *Psychological bulletin*, vol. 111, no. 1, p. 172, 1992.

[25] J. Bromley, J. W. Bentz, L. Bottou, I. Guyon, Y. LeCun, C. Moore, E. Säckinger, and R. Shah, "Signature verification using A "siamese" time delay neural network," *IJPRAI*, vol. 7, no. 4, pp. 669–688, 1993.

[26] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012, pp. 1106–1114.

[27] S. Kiranyaz, T. Ince, R. Hamila, and M. Gabbouj, "Convolutional neural networks for patient-specific ECG classification," in *EMBC*, 2015, pp. 2608–2611.

[28] M. Lin, "Network in network," *arXiv preprint arXiv:1312.4400*, 2013.

[29] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *CoRR*, vol. abs/1609.02907, 2016.

[30] W. Sheng and X. Liu, "A genetic k-medoids clustering algorithm," *Journal of Heuristics*, vol. 12, pp. 447–466, 2006.

[31] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability/University of California Press*, 1967.

[32] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *CVPR*, 2005, pp. 539–546.

[33] M. Kalisch and P. Bühlmann, "Estimating high-dimensional directed acyclic graphs with the pc-algorithm," *J. Mach. Learn. Res.*, vol. 8, pp. 613–636, 2007.

[34] P. Berkhin, "A survey on pagerank computing," *Internet mathematics*, vol. 2, no. 1, pp. 73–120, 2005.

[35] S. Park, W. Lee, B. Choe, and S.-G. Lee, "A survey on personalized pagerank computation algorithms," *IEEE Access*, vol. 7, pp. 163 049–163 062, 2019.

[36] S. Huang, Z. Wang, X. Zhang, Y. Tu, Z. Li, and B. Cui, "DBPA: A benchmark for transactional database performance anomalies," *SIGMOD*, vol. 1, no. 1, pp. 72:1–72:26, 2023.

[37] F. Karim, S. Majumdar, H. Darabi, and S. Harford, "Multivariate lstm-fcns for time series classification," *Neural Networks*, vol. 116, pp. 237–245, 2019.

[38] H. I. Fawaz, B. Lucas, G. Forestier, C. Pelletier, D. F. Schmidt, J. Weber, G. I. Webb, L. Idoumghar, P. Muller, and F. Petitjean, "Inceptiontime: Finding alexnet for time series classification," *DMKD*, vol. 34, no. 6, pp. 1936–1962, 2020.

[39] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *NIPS*, 2017, pp. 5998–6008.

[40] C. Geng, S. Huang, and S. Chen, "Recent advances in open set recognition: A survey," *TPAMI*, vol. 43, no. 10, pp. 3614–3631, 2021.

Authorized licensed use limited to: University of Electronic Science and Tech of China. Downloaded on October 13,2026 at 12:18:11 UTC from IEEE Xplore. Restrictions apply.