Unsupervised Contextual Anomaly Detection for Database Systems

Sainan Li[1,†] Qilei Yin[1,†] Guoliang Li[1] Qi Li[1] Zhuotao Liu[1] Jinwei Zhu[2]
[1]Tsinghua University and BNRist [2]Huawei

ABSTRACT

Abnormal data access operations in database systems always happen, which are typically incurred by misoperations or attacks, though these systems are enforced with strict access control policies. However, prior arts only focus on detecting abnormal data accesses by utilizing known attack patterns or identifying behaviors significantly deviated from normal behaviors. They cannot capture stealthy abnormal data access operations that are similar to normal ones. In this paper, we propose a novel unsupervised anomaly detection system UCAD, which aims to detect abnormal data access operations, by comparing operation's semantics with their *contextual intent*. However, it is non-trivial to obtain accurate semantics of operations for intent analysis because (i) the same operation may exhibit diverse semantics under different operation contexts and (ii) different operation sequences could have identical semantics due to heterogeneous user access patterns. To address this issue, we develop a new transformer model called Trans-DAS for UCAD. Trans-DAS learns the semantics of individual operations by utilizing the attention mechanism that analyzes the relevance between any pair of operations in sequence, and captures the contextual intent of operations inferred from the contexts. Specifically, Trans-DAS utilizes a particular embedding layer to embed the semantics of individual operations without the operation order information and a masking mechanism that allows Trans-DAS to learn the semantics according to the bidirectional contexts. Also, we define a new training objective for Trans-DAS to enlarge the difference among the embedded semantics. Furthermore, in order to effectively utilize Trans-DAS for detection, we develop two modules in UCAD, *i.e.,* a data preprocessing module that allows Trans-DAS to accurately learn the normal semantic information by removing noisy data, and an anomaly detection module that learns the semantic information for intent comparison. We evaluate the performance of UCAD on real-world data traces under different settings (e.g., varied parameters and hybrid datasets). The results demonstrate that UCAD achieves the average F1-score of 0.94 in two scenarios, which significantly outperform baselines, and shows robustness to hybrid data and good transferability to different tasks.

[†]These authors contributed equally to this work.

SIGMOD '22, June 12–17, 2022, Philadelphia, PA, USA.

ACM ISBN 978-1-4503-9249-5/22/06…$15.00
https://doi.org/10.1145/3514221.3517861

CCS CONCEPTS

• **Information systems** → **Database administration**; • **Computing methodologies** → *Anomaly detection.*

KEYWORDS

Anomaly Detection; Database Management; Attention Mechanism

ACM Reference Format:
Sainan Li, Qilei Yin, Guoliang Li, Qi Li, Zhuotao Liu, & Jinwei Zhu. 2022. Unsupervised Contextual Anomaly Detection for Database Systems. In *Proceedings of the 2022 International Conference on Management of Data (SIGMOD '22), June 12–17, 2022, Philadelphia, PA, USA.* ACM, NY, NY, USA, 15 pages. https://doi.org/10.1145/3514221.3517861

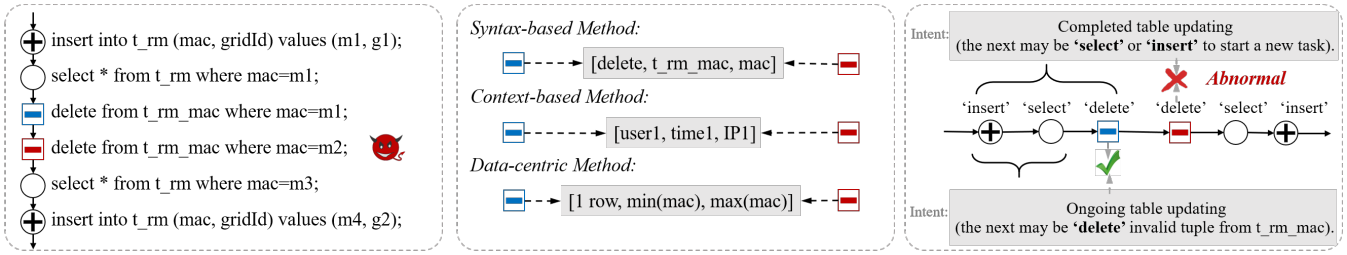
1 INTRODUCTION

The modern database systems store huge amounts of proprietary data for numerous applications. Thus, it is critical to ensure system confidentiality and integrity. Although these systems are typically protected by various peripheral defenses (*e.g.,* by enforcing access control policies or enabling intrusion detection), abnormal data access operations[1] are still possible due to accidental misoperations or deliberated attacks by sophisticated attackers. These abnormal data accesses expose database systems to a diverse range of security threats and may result in catastrophic consequences, such as data tampering and data breaches. For example, it has been reported that the economic loss caused by data breaching in the United States alone is as high as 8.64 million dollars in 2020, and the cumulative attack duration is around 280 days [3].

A series of anomaly detection methods specific for database systems have been developed to address this issue. They can be roughly classified into four categories: syntax-based methods [14, 39, 40, 64] focus on detecting anomalies by pattern matching or machine learning algorithms according to the syntax of SQL statements, context-based methods [7, 25, 42, 83] detect abnormal behaviors deviated from normal patterns by utilizing system and user information generated during data accessing, data-centric methods [43, 51] perform detection by leveraging the statistics of queried data incurred by obvious data change, and hybrid methods [52, 65, 66, 71] combining the design primitives of different methods.

However, these traditional methods focus mostly on detecting known attacks or anomalous data accesses [42, 51, 52, 61, 64, 80] whose behavior are significantly deviated from normal patterns. When these anomalous data accesses are more stealthy, *e.g.,* the attacker only launches a small amount of anomalous database operations intermittently and changes confidential data slightly, the traditional methods become less effective. Yet these stealthy

[1]A data access operation in database system refers to an individual SQL statement and we use "operation" and "statement" interchangeably in this paper.



(a) A session having one stealthy abnormal operation. **(b) The limitation of traditional detection methods.** **(c) Our key observation about abnormal operations.**

Figure 1: A data access example containing one abnormal delete operation (in red). Traditional methods generate indistinguishable features for both normal and abnormal delete operations. Yet we observe that the semantics of abnormal operation is deviated from the contextual intent inferred by its preceding operations.

anomalous data accesses are common in real-world production systems [1, 2, 4, 5]. As shown in Figure 1 (a) and (b), we use a real-world example from our production system to illustrate the limitations of these traditional methods. An attacker uses a user's legitimate credential (user1) and address (IP1) to remotely access the database and performs several ordinary operations for table updating during the period of time1. Meanwhile, the attacker stealthily deletes (in red) important data from table t_rm_mac after a normal delete (in blue) operation on table t_rm_mac. The abnormal operation cannot be captured by these methods since they generate *indistinguishable* feature vectors for both normal and abnormal delete operations. For example, the syntax-based methods extract the same feature vector from their syntax, *i.e.,* the same command type on the same table and column. The context-based approaches cannot find their difference as the user attributes (user account, access time, and client address) are not changed. The data-centric methods analyze that both delete operations only remove one row of data and the statistic features, *i.e.,* the min, max values of the targeted column, are the same.

To address the above problem, we propose a novel unsupervised anomaly detection system, Unsupervised Contextual Anomaly Detection (UCAD). Our key observation underpinning the design is that: *abnormal operations can be identified by comparing the semantics of an individual operation when considered alone with the operation's contextual intent, obtained by learning the semantics of a sequence of operations preceding*[2] *the operation in question.* Consider the following intuitive example illustrated in Figure 1 (c). When analyzing the first normal delete, its former insert and select operations reveal that the intent is an ongoing table updating, so that the next operation is likely to delete an invalid tuple due to the newly inserted data. For the second abnormal delete, its former insert, select, and the first delete operations indicate that the intent is a completed table updating, which means that the next operation should be select (*i.e.,* to start a data query task) or insert (*i.e.,* to start another table updating task). Yet, in this example, the attacker performs another delete (in red) in an attempt to stealthily sabotage other data, which is deviated from the intent of a sequence preceded this operation.

Challenges. Nevertheless, to accurately capture the semantics of data access operations, we need to address three challenges. First, *the same operation may exhibit diverse semantics under different operation contexts.* For instance, the two identical delete statements in Figure 1 indicate distinct semantics as their preceded operation sequence are different. Thus, the common semantics extraction approaches (*e.g.,* word embedding [27, 55]) are ill-suited for this task since they can only learn fixed semantic representations from local contexts. Second, *different operation sequences could have identical semantics due to heterogeneous user access patterns*, which means that the order of operations is not sufficient or even misleading in capturing the semantics of users' operation sequence (*i.e.,* the intent). However, traditional sequence models like LSTM rely heavily on the order information for semantics extraction. Thus, they are not applicable in database systems with heterogeneous user behaviors. Third, *noise is non-negligible in raw data access operation records.* In database systems, operations that are irrelevant to the true intent in question are common, *e.g.,* accidental misoperations (not necessarily malicious). These "noisy" data may interfere with the process of extracting the semantics of data access operations.

Contributions. In this paper, we propose a novel unsupervised anomaly detection system UCAD to identify abnormal data access operations. To solve the above challenges and obtain accurate semantics of operations, we develop a new transformer model called Trans-DAS for UCAD. Trans-DAS utilizes a particular embedding layer to embed the semantics of individual operations without operation ordering information, and a masking mechanism that allows Trans-DAS to learn the semantics according to the bidirectional contexts. Also, we define a new training objective for Trans-DAS to enlarge the difference among the embedded semantics such that Trans-DAS can easily capture the abnormal operations. Moreover, in order to effectively detect abnormal operations using Trans-DAS, we design a preprocessing module in UCAD to filter the noisy data and known attacks, and utilize a clustering method to balance semantic patterns and remove the sessions deviated from common user behaviors. Note that, since it only requires normal data access information to learn the semantics of operations, our system works in an unsupervised manner.

In summary, we make the following contributions in this paper.

- We propose a novel unsupervised anomaly detection system UCAD to identify the stealthy abnormal data access operations in database systems.
- We develop a new transformer model called Trans-DAS for UCAD to capture the semantics and contextual intent of operations.

[2]During the model training process, we can potentially use the operations after the operation in question, *i.e.,* a sequence surrounding the operation in question.

- We prototype UCAD. Particularly, UCAD includes a preprocessing module that allows Trans-DAS to obtain normal semantic information by removing noisy data, and an anomaly detection module that implements an instance of Trans-DAS to detect abnormal operations via contextual intent comparison.
- We perform extensive evaluations for UCAD using two real-world data traces in two typical data access scenarios. The experimental results show that it can achieve the F1-score of 0.89693 and 0.98168 in two scenarios, respectively, which significantly outperform baselines. Also, the results demonstrate that Trans-DAS is not sensitive to different settings and robust to abnormal training data. Furthermore, the evaluations on three public datasets demonstrate the transferability of UCAD to other tasks[3].

2 THREAT MODEL

In this paper, we consider the abnormal data access operations that are able to breach peripheral protections of database systems, such as stealing legitimate credentials [80] or misoperations. In particular, these data access anomalies can occur due to the following reasons.

Privilege Abuse. Authorized users abuse their privileges to perform abnormal operations intentionally [52, 61, 64], *e.g.,* for the purpose of personal financial incentives. For example, an attacker can always perform more query operations to retrieve confidential data violating normal business rules, and even delete data to sabotage the database systems [52].

Credential Stealing. An attacker steals credentials of legitimate users to access database and then stealthily performs abnormal data access operations [42, 51, 80]. Generally, abnormal operations are hidden deeply in the disguise of numerous normal daily activities [42], *e.g.,* an abnormal delete operation hidden in a session to remove confidential and sensitive data.

Misoperations. An inexperienced staff may perform misoperations accidentally, resulting in data chaos such as data leakage. Compared with normal data access operations, their operations are not logically consistent and considered abnormal [52].

Note that, in this paper, we assume that operations of each user are correctly recorded in the database system log. Attackers are unable to corrupt the integrity of system log, which is our trusted computing base (cf., [21, 24, 47, 54, 70, 81]). For instance, they cannot execute abnormal SQL statements without generating any log in the database system. An sophisticated attacker may tamper or even remove their operation log by exploiting vulnerabilities of database system, which is beyond the scope of this paper and can be addressed by existing memory space protection techniques [15, 16].

3 OVERVIEW OF UCAD

Figure 2 shows an overview of our unsupervised anomaly detection system UCAD. Architecturally, UCAD consists of a preprocessing module and an anomaly detection module. The preprocessing module tokenizes the raw data access operations in system log and remove the noisy data so that the anomaly detection module can learn the semantic information accurately. The anomaly detection module implements an instance of Trans-DAS, which learns the semantic information for contextual intent comparison.

[3]The source code is released in https://github.com/UCAD3/core.

In general, UCAD has two working stages: *Offline Training* and *Online Detection*. In the training stage, the preprocessing module builds the vocabulary for data access operation tokenization, and removes noisy session data according to tokenized keys in the session so that we can obtain a purified training set containing normal user sessions[4]. The anomaly detection module trains Trans-DAS on the purified dataset to learn the normal semantic information. Note that this training procedure can be periodically conducted or manually triggered to make Trans-DAS capture the latest normal semantic information. In the detection stage, the preprocessing module utilizes the learned vocabulary to tokenize each active user session and directly filters out the known attack patterns. The anomaly detection module utilizes the trained Trans-DAS model to evaluate whether the semantics of current operation in the active user session matches the overall intent of a sequence of operations preceding the operation in question (which we refer to as *contextual intent of the operation*). Since the normal semantic information is learned via Trans-DAS, Trans-DAS can capture the anomalies that do not include normal semantic information, *i.e.,* a mismatched operation is labeled as anomaly. The detected abnormal operations may be subsequently sent to a domain expert for further investigation and actions (such as banning the user or even re-qualify the system software).

4 THE TRANS-DAS MODEL

In this section, we describe our transformer model Trans-DAS that used in UCAD for anomaly detection.

4.1 Basic Idea

To capture the semantic information of operations, we develop a new transformer model called Transformer for Data Access Semantics (Trans-DAS). The goal of Trans-DAS is to capture the relevance between one operation and its bidirectional operation contexts so that Trans-DAS can learn the exact semantics of individual operations and the contextual intent.

Note that, traditional models widely used in semantics extraction, *i.e.,* the Long Short Term Memory (LSTM) network and traditional Transformer models, are ill-suited for learning the semantic information from data access operations. Specifically, although LSTM network has been a standard solution to learn semantic patterns from sequential data, they process data based on the item order in the sequence. Such processing implicitly makes LSTM relies heavily on the order information (*i.e.,* the order dependence) to learn semantics, which is not applicable in database systems since heterogeneous user access patterns exhibit diverse operation sequences. Moreover, the traditional transformer model [79], consisting of an encoder and a decoder, learns the semantic information by using the attention mechanism that captures the relevance between each pair of items. However, it also embeds the position encodings (*i.e.,* order information) into the semantics of items. When facing the challenge of heterogeneous access patterns, such order information may prevent us from capturing the semantic information accurately. Furthermore, the encoder chooses a fully-connected attention design without masking (*i.e.,* connecting an item with

[4]A user session refers to a sequence of data access operations executed by a specific user during one time of database accessing.

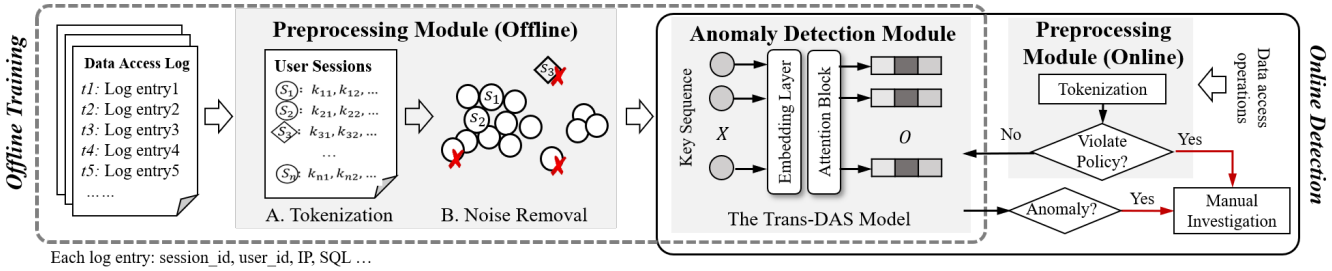


Figure 2: The overview of our anomaly detection system (UCAD) which consists of a preprocessing module and an anomaly detection module. UCAD has two working stages: offline training and online detection.

all items including itself). Thus, capturing the semantics of an operation might be influenced by the operation itself. The decoder adopts a *masking mechanism* that only connects an operation with its preceded operations, resulting in learning partial contextual intent. Thus, the encoder and the decoder represent a weak capability in learning the semantic information of data access operations.

Our Trans-DAS model well addresses the issues above. As shown in Figure 3 (a), it consists of an embedding layer (see Section 4.2) and multiple attention blocks (see Section 4.3). The embedding layer is used to embed the semantics of individual operations without the order information. The attention block uses a new masking mechanism, which connects an operation with its bidirectional operation contexts except itself. The masking mechanism prevents inferring the semantics of an operation directly from itself and allows Trans-DAS to capture the contextual intent of an operation based on its bidirectional contexts. As Trans-DAS is designed for unsupervised detection, the input is only a sequence containing n operations and the output is a predicted sequence of the same length, which can be used for anomaly detection.

4.2 The Embedding Layer

The embedding layer is designed to convert each operation into a vector representing the latent semantics. Suppose each operation in the input sequence can be identified by a unique key and h is the hidden dimension, the embedding layer builds a matrix $\mathbf{M} \in \mathbb{R}^{n \times h}$ during model training and then each operation key can be converted into an embedding vector by retrieving $\mathbf{M}$. And a constant zero vector $\mathbf{0}$ is used to represent a specific key which is preserved for padding and new operations appeared during detection. Through the embedding layer, an input sequence of length L can be represented as:

$$\mathbf{E} = (\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_L), \quad \mathbf{e}_i \in \mathbb{R}^h, \tag{1}$$

Compared with the traditional transformer model, we remove the position encoding of each item (*i.e.,* the order information) from embedding vectors, to get rid of the order dependence. Thus, we can obtain accurate semantic representations under the heterogeneous user access patterns.

4.3 Multiple Attention Blocks

To learn the semantics of operation sequence (*i.e.,* the contextual intent), we design an attention block and its architecture is shown in Figure 3(b). An attention block consists of a multi-head attention layer, *i.e.,* self-attention layers in parallel, and a feed forward layer. Note that the self-attention layer utilizes the self-attention mechanism [79] and a masking mechanism of our own design. We describe their details as follows.

The Multi-Head Attention Layer with Masking. The attention mechanism can be described as a function that maps a *query* and a set of *key-value* pairs to an output, where the *query*, *key*, *value* are all vectors projected from input data individually [79]. The output is another vector that is computed as a weighted sum of the *values*, where the weight of each *value* is computed by a compatibility function of the *query* with the corresponding *key*. When the attention mechanism is applied to correlate different parts of a single sequence in order to compute a new representation of the same sequence that reveals its internal relevance, *e.g.,* converting the operation sequence into semantic representations, the mechanism can be called self-attention [79]. In particular, given an embedded input sequence $\mathbf{E} \in \mathbb{R}^{L \times h}$, the self-attention layer (**SA**) linearly projects $\mathbf{E}$ into three individual projections representing query, key and value using three learnable weight matrices, and then feeds them into an *attention* function together:

$$\mathbf{SA}(\mathbf{E}) = \mathbf{Attention}(\mathbf{E}W^Q, \mathbf{E}W^K, \mathbf{E}W^V), \tag{2}$$

where $W^Q, W^K, W^V \in \mathbb{R}^{h \times h}$ are weight matrices for query, key and value projections, respectively. And we apply the *scaled dot-product attention* [79] function:

$$\mathbf{Attention}(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{h}}\right)V, \tag{3}$$

In general, it computes dot products of each query with all keys, scales the outputs by dividing $\sqrt{h}$, and applies a softmax function to obtain the weights on values. These weight matrices represent the semantic relevance among operation contexts.

A common unsupervised training strategy for learning semantics is applying the next prediction task, *i.e.,* the target output for an input sequence $(x_{t-n+1}, ..., x_{t-1}, x_t)$ is a shifted version $(x_{t-n+2}, ..., x_t, x_{t+1})$. This strategy makes learning-based model capture the semantic information by predicting an operation from its contexts. However, in this strategy, the i-th target operation in the output sequence is the same as the $(i{+}1)$-th operation in the input sequence. If we choose a fully-connected attention design as the encoder of transformer, the prediction of an operation will be influenced by itself. Besides, the future masking mechanism in the transformer's decoder only uses unidirectional items before the $(i{+}1)$-th input to predict the i-th output. Neither design is able to capture the contextual intent accurately. To overcome their limitations, we design a new masking

mechanism for our self-attention layer, which is realized by disconnecting Q_i and K_{i+1} so that the influence from operation itself is eliminated. Figure 3 (c) shows a simplified schematic of this design. In summary, our self-attention layer uses bidirectional operation contexts (only except the (i+1)-th input) to predict i-th output, so that Trans-DAS can learn the semantics of individual operations and contextual intent accurately.

Since each self-attention (*i.e., single-head attention*) layer can profile a specific kind of semantic pattern, we parallel multiple self-attention layers to compose a *multi-head attention* layer (**MH**). Thus, Trans-DAS can capture diverse semantic patterns. Formally:

$$\mathbf{MH}(\mathbf{E}) = [\mathbf{SA}_1(\mathbf{E}); \mathbf{SA}_2(\mathbf{E}); ...\mathbf{SA}_m(\mathbf{E})]\, W^O, \tag{4}$$

where the outputs from m heads are concatenated together and then projected with $W^O \in \mathbb{R}^{h\times h}$. Note that the current projection matrices are $W_i^Q, W_i^K, W_i^V \in \mathbb{R}^{h\times h/m}$ in each $\mathbf{SA}_i$. As the dimension of each head is reduced from h to h/m, the computation cost is similar to that of a single-head attention with full dimensions.

We apply three regularization techniques: (1) *Residual connection.* It propagates the intermediate output of lower layers to higher layers through skip connections, to avoid the problem of gradient vanishing. (2) *Layer normalization.* It is beneficial for enhancing the model's generalization ability. (3) *Dropout.* It is effective to alleviate the overfitting problem. The whole regularization process is:

$$\mathbf{Reg}(\mathbf{x}) = \mathbf{LN}(\mathbf{x} + Dropout(f(\mathbf{x})), \tag{5}$$

$$\mathbf{LN}(\mathbf{x}) = \frac{\mathbf{g}}{\sqrt{(\sigma)^2 + \epsilon}} \odot (\mathbf{x} - \mu) + \mathbf{b}, \tag{6}$$

where $f(\mathbf{x})$ represents the output of the multi-head attention layer, $\mathbf{g}$, $\mathbf{b}$ are the gain and bias parameters, μ, σ are the mean and the variance of $\mathbf{x}$. $\odot$ is the element-wise multiplication between two vectors. ϵ is a small constant to prevent division by zero. The output of this multi-head attention layer is $\mathbf{Reg}(\mathbf{x})$.

The Feed Forward Layer. After the multi-head attention layer, we append a point-wise feed-forward layer to each output position separately and identically, to further enhance the capability of semantic representation. It refers to two linear transformations with the ReLU function:

$$\mathbf{FFN}(x) = max(0, xW_1 + b_1)W_2 + b_2, \tag{7}$$

where $W_1, W_2 \in \mathbb{R}^{h\times h}$, $b_1, b_2 \in \mathbb{R}^h$ are also learnable parameters and shared among all positions. We apply the same regularization techniques used in multi-head attention layer, *i.e.,* $\mathbf{FFN}(x)$ is regarded as $f(\mathbf{x})$ in Equation 5 to compute the regularized output.

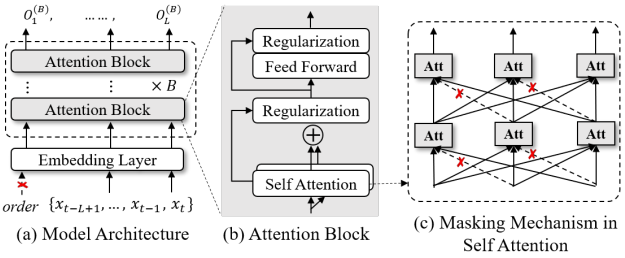
Stacked Attention Blocks. Finally, we stack attention blocks to further strengthen the capability of Trans-DAS so that Trans-DAS can capture complex semantic patterns. Therefore, the output of the b-th block can be obtained as follows:

$$I^{(b)} = \mathbf{MH}(O^{(b-1)}), \quad O^{(0)} = \mathbf{E}, \tag{8}$$

$$O_i^{(b)} = \mathbf{FFN}(I_i^{(b)}), \quad i \in [1, L], \tag{9}$$

5 THE DESIGNS OF UCAD

In this section, we present the designs of the key modules of UCAD, *i.e.,* a data preprocessing module (see Section 5.1) for efficient Trans-DAS training and anomaly detection, and an anomaly detection module by utilizing Trans-DAS. In particular, the anomaly detection module offline trains Trans-DAS (see Section 5.2) and online detects anomalies by using trained Trans-DAS (see Section 5.3).



(a) Model Architecture (b) Attention Block (c) Masking Mechanism in Self Attention

Figure 3: A detailed view of Trans-DAS. Compared with the original Transformer, Trans-DAS uses an embedding layer without order information, and multiple attention blocks utilizing a new masking mechanism which takes in bidirectional contexts, to capture the semantics of individual operations and the contextual intent.

5.1 The Preprocessing Module

The preprocessing module is used to: (i) tokenize the raw data access log of normal users and (ii) remove noisy data to purify the training data for learning accurate semantic information. In particular, it processes the raw data access log by the following two steps: *operation tokenization* and *noise removal.*

Operation Tokenization. We first tokenize each operation (*i.e.,* SQL statement in plain text) in the raw data access log as a unique quantitative value, *i.e.,* the statement key[5]. Specifically, we abstract all variables in each statement as "$" to protect user privacy. For instance, the statement "Update T_content set count=23 where danmuKey=94" will be abstracted as "Update T_content set count=$1 where danmuKey=$2". We take a concise yet effective tokenization method that assigns a unique key starting from "k_1" to a specific statement, while "k_0" is preserved for padding and newly appeared statements during detection.

Compared with traditional tokenization methods [20, 24, 75], our design can distinguish fine-grained differences in SQL statements, which may result in significant differences in their semantics. For example, "delete from t_mac where normal_mac=$1", and "delete from t_mac where abnormal_mac=$1" are literally similar yet semantically inconsistent. The methods utilizing only the *main* part of the statement ignore such a small literal difference and assign the same key to these two statements, *e.g.,* [20] extracts the longest common subsequence among statements. Thus, their capability of capturing the diverse semantics of operations is insufficient. By contrast, our design will assign two unique keys to these two statements, respectively. After this step, each user session will be converted into an operation key sequence. Note that the built vocabulary for operation tokenization is preserved to tokenize active user sessions during the detection stage.

Noise Removal. Since the noise in raw data access records is non-negligible, it is necessary to purify user sessions for Trans-DAS training. Although Trans-DAS is resilient to a certain amount of noise in the training data (see Section 6.5), clean data is still

[5]For short, we use "key" to denote "statement key" in the rest of paper.

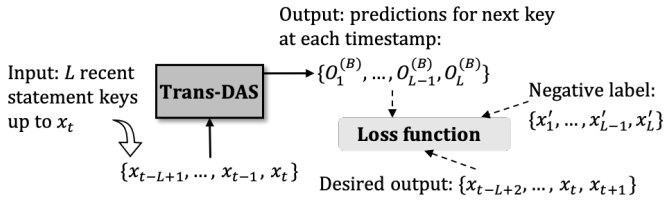


Figure 4: The training stage of Trans-DAS model.

helpful to capture more accurate normal semantic information, yielding better detection performance. To remove noise, we apply: (1) enforcing access control policies; (2) clustering.

First, we utilize the attribute-based access control policies [34] to directly filter out the known attack patterns. Based on existing studies [6, 33, 35, 38], we use the identity of user, the access address, the access time, the target access table, and the interval between two consecutive operations in a session, to establish access control policies. Then the sessions violating the granting policies or matching the denying policies are directly filtered out. For instance, we can use the user identity and the address attributes to construct specific policies restricting the users who access the database from previously unknown addresses, since an unknown address is a typical characteristic of data access anomaly [6]. Note that the access control policies are extensible, so that we can easily introduce new policies to filter out more known attack patterns.

We then apply clustering method to remove noisy sessions which are deviated from common data access patterns. Specifically, we profile each session (*i.e.,* key sequence) by using the n-gram features [44, 56, 76, 78], compute the similarities among sessions by Jaccard Index and perform clustering based on the DBSCAN algorithm [26], which is capable of discovering clusters with arbitrary shapes. Based on the clustering results, we take the following steps: (1) We perform randomly under-sampling on large clusters for balancing data access patterns, where the sampling rate is decided by the median size of all clusters. It will prevent the normal yet relatively less patterns (in small clusters) from being wrongly regarded as noise by Trans-DAS. (2) After balancing, we remove the clusters whose size is significantly smaller than the median size of all clusters, as their data access patterns are rare. (3) We check the average length (*i.e.,* the number of key) of sessions in each cluster, and remove sessions whose length is much smaller than the average or median length value. These small sessions are too short to reveal the contextual intent of operations and may even contain ambiguous semantics. Finally, we regard the user sessions in remaining clusters as purified training data for anomaly detection module.

5.2 Offline Trans-DAS Training

We now describe the offline training of Trans-DAS in the anomaly detection module. To prepare normal training data, the preprocessing module converts raw data access log into user sessions and filter out the noisy data as discussed in Section 5.1. Then, the anomaly detection module offline trains Trans-DAS on the purified normal sessions to capture the semantic information.

As shown in Figure 4, Trans-DAS is trained in an unsupervised manner, which uses a shifted version of the input sequence as its desired output sequence. Formally, assuming there is a normal training dataset $\mathcal{T} = (S_1, S_2, ...S_N)$ containing N normal sessions, each session represents a sequence of operation keys chronologically executed by a database user. For each $S_i \in \mathcal{T}$, we extract operation sequences as the input of Trans-DAS using a sliding window of size L. For an extracted sequence $(x_{t-L+1}, ..., x_{t-1}, x_t)$, its desired output key sequence is $(x_{t-L+2}, ..., x_t, x_{t+1})$. Our training objective is to maximize the similarity between the output of Trans-DAS and the desired key sequence.

To achieve this goal, we first convert the desired key sequence into a semantic embedding vector by retrieving matrix $\mathbf{M}$ in the embedding layer of Trans-DAS, then obtain the last layer's output: $O^{(B)} = (O_1^{(B)}, O_2^{(B)}, ..., O_L^{(B)})$. The similarity between $O_i^{(B)}$ and an operation key x_j is defined as their inner product:

$$z_i^j = Sigmoid\left(O_i^{(B)} \cdot \mathbf{M}\left(x_j\right)\right), \tag{10}$$

We develop a new loss function consisting of three components for Trans-DAS to address the challenge of diverse operation semantics. We should enlarge the difference among different operations semantics such that Trans-DAS can capture them more accurately. To achieve this goal, we utilize *triplet loss* [68] as our first component. It tries to maximize the distance of two embeddings with different labels and minimize that of two embeddings with the same label so that the difference between embeddings (*i.e.,* the embedded semantics) could be enlarged. In addition, to make Trans-DAS output the desired operation sequence, we use the one-class *cross entropy loss* that computes the absolute semantic distance between the predicted and desired operations as our second component. Finally, to avoid overfitting, we use the L_2 normalization as the third component. As a whole, our loss function is defined as:

$$\mathcal{L} = \sum_{\mathcal{T}} \sum_{i=1}^{L} \max\left(z_i^- - z_i^+ + g, 0\right) - \log\left(z_i^+\right) + ||\theta||_2, \tag{11}$$

where g is a margin parameter, z_i^+ represents the similarity between $O_i^{(B)}$ with its desired operation, and z_i^- refers to the similarity between $O_i^{(B)}$ with its undesired operation. Note that the desired operations can be achieved by forward shifting the input operation sequence. The choice of undesired labels for each session is based on the concept of Negative Sampling [27]. Namely, we choose random operation keys that never appear in S_i as the negative labels iteratively. Next, we can train Trans-DAS with common optimization methods like SGD.

We apply a concise yet effective fine-tuning strategy [49] to solve the issue of *concept drift* [49, 77], which is a common obstacle for learning-based systems. It is typically caused by the variation of data patterns over time and may lead to false alarms. Specifically, when our system finishes model training and starts to perform online detection, we collect new normal sessions verified by our system. In the next round of training, we use these sessions to fine-tune our previously trained Trans-DAS. Our strategy achieves advantages over other approaches including retraining a new model on new data or producing ensemble models with the original models [49]. In particular, it retains the information of historical data for detection and models the normal semantic patterns by utilizing the data. However, retraining a new model from scratch is often

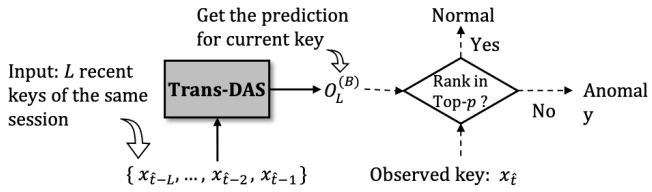


Figure 5: The detection stage of Trans-DAS model.

constrained by the availability of new data, and may even fail to capture the historical semantic patterns. Moreover, we only fine-tune one deep learning model, and therefore imposes much less storage and computation compared with training ensemble models.

5.3 Online Detection with Trans-DAS

Now anomaly detection module detects anomalies by utilizing the trained Trans-DAS. Similar to the training stage, the preprocessing module first tokenizes data access operations of active user sessions into keys. Second, the trained Trans-DAS model detects abnormal data access operations by validating whether the semantics of current operation in the active session matches the overall intent of its preceding operation sequence.

Recall that, during the training process of our Trans-DAS model, the last embedding vector in the output sequence represents the contextual intent of current operation predicted by its preceding operations. Intuitively, if the actually observed operation is inconsistent with the predicted contextual intent, it can be regarded as an anomaly. However, this exact matching strategy may yield plenty of false alarms in practice. As discussed in Section 1, the user access pattern is heterogeneous so that there may exist multiple legitimate options for the next operation. For instance, after inserting tuples into a table, the normal user may perform following operations: (a) select from the same table to ensure the insert operation is successful; (b) insert tuples into other tables; (c) delete invalid data in another table due to the former insert operation.

To address this issue, we develop a detection mechanism based on the *top-p* strategy as illustrated in Figure 5. Specifically, for an active user session, we feed its preceding operation key sequence $\widehat{S} = (x_{\hat{t}-L}, ..., x_{\hat{t}-1})$ into Trans-DAS at time $\hat{t}$, and obtain the last embedding vector of model output ($O_L^{(B)}$), which contains the normal contextual intent of operation at time $\hat{t}$. Then, we compute the similarities between $O_L^{(B)}$ and all operations individually via Equation 10 and rank these results from largest to smallest. With the ranking list, we check whether the similarity between real operation $x_{\hat{t}}$ and $O_L^{(B)}$ ranks top-p. If so, we think $x_{\hat{t}}$ is normal as it matches the contextual intent. Otherwise, $x_{\hat{t}}$ is regarded as an abnormal data access operation and this user session is immediately flagged as an abnormal session. Finally, a domain expert is responsible for diagnosing sessions that contain detected abnormal operations. The false alarms will be incorporated with the verified normal sessions for the next round of Trans-DAS training.

6 EVALUATION

In this section, we evaluate UCAD with real-world data traces around the following questions.

RQ1: *By analyzing the semantics of operations, can UCAD accurately identify abnormal data access operations and outperform existing approaches?*
RQ2: *Comparing with the original Transformer model [79], can our new designs in Trans-DAS improve the detection performance of UCAD?*
RQ3: *Is the performance of UCAD sensitive to hyper-parameters? And is it possible to reduce the training time while retaining good detection capabilities?*
RQ4: *If the training set of Trans-DAS does not only contain normal data, can UCAD still perform accurate detection based on the anomaly detection module?*
RQ5: *Can UCAD be effectively transferred to relevant tasks?*
RQ6: *Can UCAD identify real-world anomalies?*

6.1 Experimental Setup

Dataset. We collect data about two typical database application scenarios from an Internet company. The first scenario is online commenting applications where users perform more insert, delete, and update operations. The second one supports location service applications and contains more select but less update behaviors. Table 1 summarizes our datasets used for evaluation.

For each scenario, we collect real world data and purify it via our preprocessing module. The purified normal user sessions are divided into a training set ($\mathcal{T}$) and a testing set ($\mathcal{V}1$) with a ratio of 8:2. Moreover, to simulate heterogeneous user access patterns in database, we generate two other normal datasets ($\mathcal{V}2$ and $\mathcal{V}3$) for testing as follows: (1) *Partially Swap:* Given a normal session in $\mathcal{V}1$, we create a session in $\mathcal{V}2$ by choosing the partially interchangeable operations in the session, randomly swap them, and *manually verify* that the swapping does not undermine the session goal or change its characteristics. For instance, several consecutive select operations querying different tables in the same session are perfect candidates for such mutations. (2) *Partially Remove:* Since some operations in a session are irrelevant to the session goal, *e.g.,* a user performs repeated select multiple times, we remove part of these operations in the original session in dataset $\mathcal{V}1$, and also *manually verify* that both the session goal and its normal characteristic are not changed, to create a new session in $\mathcal{V}3$.

Due to the rarity of anomalies in realistic scenarios, we can only collect a very small amount of abnormal sessions. Meanwhile, to the best of our knowledge, there is no public dataset about abnormal database operation log. Hence, we synthesize abnormal data based on the methods used in [42, 52], to simulate the three kinds of anomalies introduced in Section 2. Specifically, (1) For the privilege abuse, we carefully prepare a rich set of select operations and generate each abnormal session by combining repeatedly or randomly chosen select operations with a normal session from $\mathcal{V}1$. (2) For the credential stealing, we randomly insert delete and other irrelevant operations into a normal session of $\mathcal{V}1$ to generate a new abnormal session. The amount of newly inserted operations is less than 10% of the original operations, to ensure the abnormal behavior is stealthy. (3) For the misoperations, we select the normal operations which are rarely performed and randomly combine multiple of them to construct an abnormal session. We denote the abnormal sets generated by three kind of attackers as $\mathcal{A}1$, $\mathcal{A}2$ and $\mathcal{A}3$, respectively, and the size of each set is the same to that of $\mathcal{V}1$.

Table 1: Our training and testing datasets collected from two database application scenarios. #Training session and #Testing session are the numbers of user sessions used for training and testing, respectively.Average Length is the average number of database operations in a user session. #Keys is the number of each database operation (*i.e.,* select, insert, update and delete). #Table is the number of database tables.

	#Training session	Average length	#Keys (select, insert, update, delete)	#Table	#Testing session	
					Abnormal	Normal
Scenario-I	354	24	20 (7, 4, 4, 5)	7	89×3	89×3
Scenario-II	3722	129	593 (238, 351, 146, 4)	15	930×3	930×3

In summary, for each scenario, we use $\mathcal{T}$ as the training set and integrate the normal sets $\mathcal{V}1$, $\mathcal{V}2$, $\mathcal{V}3$ and abnormal sets $\mathcal{A}1$, $\mathcal{A}2$, $\mathcal{A}3$ as the testing dataset.

Baselines. We choose five unsupervised anomaly detection methods as baselines. First, we select the classical Tree-based Isolation Forests (iForest) [48] and Kernel-based OneClassSVM [67] algorithms due to their success in numerous applications. Since they are not specific to database systems and cannot handle time series data, we profile each session as a vector of n-dimensions (n is the number of total operation keys) and count the appearances of each operation. Second, we choose a hybrid solution [52] which utilizes statistical features. Lastly, we select the Deeplog [21] approach, which is general for anomaly detection from system log, and a state-of-the-art anomaly detection method for time series, *i.e.,* USAD [11]. Since Deeplog [21] and USAD [11][6] are capable of detecting anomalies in time series data, we directly use the same operation sequences generated by our data processing module as their training and testing sets.

Metrics. We evaluate the detection performance through three typical metrics: Precision, Recall and F1-Score. Although UCAD is designed to identify anomalies at the granularity of operation, we report detection results at the granularity of user session in order to use the same performance measure as baseline methods ([11, 48, 67] can only perform detection at the session level). Specifically, when UCAD detects an abnormal operation from a testing session, we mark this session as an abnormal session. Therefore, with the normal testing datasets (*i.e.,* $\mathcal{V}1$, $\mathcal{V}2$, $\mathcal{V}3$), we can calculate the false positive (FP), true negative (TN) and false positive rate ($FPR = \frac{FP}{FP+TN}$). Note that the normal and abnormal sessions are considered as negative and positive, respectively. On abnormal datasets (*i.e.,* $\mathcal{A}1$, $\mathcal{A}2$, $\mathcal{A}3$), the false negative (FN), true positive (TP) and false negative rate ($FNR = \frac{FN}{FN+TP}$) can be computed. Finally, we use the following three metrics: Precision=$\frac{TP}{TP+FP}$, Recall=$\frac{TP}{TP+FN}$, F1-Score=$\frac{2\times Precision\times Recall}{Precision+Recall}$.

Parameters. By default, we use the following parameter values for UCAD: $L = 30$, $p = 5$, $g = 0.5$, $h = 10$, $m = 2$, $B = 6$ in Scenario-I, $L = 100$, $p = 10$, $g = 0.5$, $h = 64$, $m = 8$, $B = 6$ in Scenario-II. Recall that L is the size of input keys to Trans-DAS, which is decided by the average length of training sessions, top-p operations (ranked by similarities) are considered as normal, g, h, m, B denote the margin parameter, the dimension of hidden vector, the number of attention head, and the number of stacked multi-head attention layers, respectively. We evaluate their impacts on the detection performance in Section 6.4. For baselines, we explore their parameter spaces and report the best results.

[6]Deeplog source code: https://github.com/wuyifan18/DeepLog; USAD source code: https://github.com/manigalati/usad

6.2 Detection Performance

Table 2 presents the performance of five baseline methods and UCAD. It is obvious that UCAD achieves the best F1-score of 0.89693 and 0.98168 in both scenarios, showing an average improvement of about 14.9% and 30.5% over the baselines.

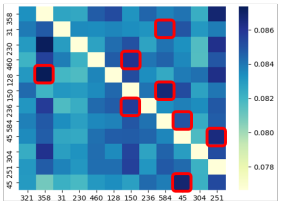
We notice that non-sequence methods [48, 52, 67] have relatively lower FPRs but higher FNRs, which means that they tend to assume sessions as normal. In particular, the FPRs of Mazzawi et al. [52] in Scenario-I are less than 0.10 while its FNR on $\mathcal{A}2$ is as worse as 1.0. Although OneClassSVM method achieves a good precision in Scenario-I, it has a poor recall of only 0.73408, and over 75% sessions in $\mathcal{A}2$ are misjudged as normal ones. This phenomenon indicates that these methods are incapable of identifying the operation sequences which contain stealthy anomalies. The main reason behind their poor performance is that they can only detect point anomalies whose statistical features are far away from the normal sessions, but ignore the semantic information. Therefore, when the attacker stealthily inserts a small amount of anomalies into normal operation sequences (*e.g.,* $\mathcal{A}2$), these methods cannot distinguish these abnormal sessions from normal ones. On the other side, the sequence methods [11, 21] achieve lower FNRs but higher FPRs, which means that they prefer to regard sessions as anomalies. For instance, DeepLog achieves FNRs of 0.01124 and 0.16022 on abnormal $\mathcal{A}2$, respectively, while its FPRs on normal testing sets are the worst (*i.e.,* 0.38202, 0.57303, 0.38202 in Scenario-I and 0.34861, 0.75591, and 0.69677 in Scenario-II). The worse performance is caused by their heavy dependence on the operation order, which is not applicable to heterogeneous user access patterns in database. Thus, they cannot capture the intent of normal operation sequence and then generate a lot of false alarms.

Equipped with several designs to accurately extract the semantic information from operation sequence, UCAD achieves lower FPRs and FNRs than other methods in the majority of cases. The only one exception is the FNR on $\mathcal{A}1$ of Scenario-I (about 0.19), which is higher than that of USAD [11] (about 0.09). This might be caused by smaller training set in scenario-I. However, note that the *average* FNRs of USAD [11] are about 0.15 and 0.11 in Scenario-I and II, respectively, while ours are only about 0.13 and 0.001. Moreover, the FPR of USAD [11] is also higher (worse) than ours on every testing dataset. Therefore, our UCAD outperforms [11] in both scenarios. Notably, the FNR of our method on the most stealthy anomaly type (*i.e.,* $\mathcal{A}2$) is only 0.02247 in Scenario-I, and it further decreases to 0.00430 in Scenario-II, which is about 20 times better than the sub-optimal result (0.08925 achieved by iForest). We argue that lower FNR is really meaningful since even a single abnormal session may cause catastrophic consequence on the database system, especially when the database stores confidential and sensitive data.

In Trans-DAS, the semantics of individual operation is inferred by its operation contexts and their attention weights (see Equation 3). We visualize the attention weights in Trans-DAS to illustrate this critical process. Since the semantic information becomes more implicit in deeper attention blocks, we choose to visualize the attention weights in the first attention block. As shown in Figure 6,

Table 2: Performance comparison in Scenario-I and II. Metrics are false positive rate (FPR), false negative rate (FNR), precision (P), recall (R) and F1-score (F1). The lower FPR and FNR means better, higher P, R and F1 means better.

Scenario	Methods	FPR			FNR			P	R	F1
		$\mathcal{V}1$	$\mathcal{V}2$	$\mathcal{V}3$	$\mathcal{A}1$	$\mathcal{A}2$	$\mathcal{A}3$			
I Commenting Application	OneClassSVM[67]	0.02247	0.02247	0.02247	0.0494	0.75281	0.0	**0.97029**	0.73408	0.83582
	iForest [48]	0.26966	0.26966	0.22472	0.20225	0.19101	0.0	0.77333	0.86891	0.81834
	Mazzawi et al. [52]	0.05618	0.05618	0.07865	0.44944	1.0	0.0	0.89032	0.51685	0.65403
	DeepLog [21]	0.38202	0.57303	0.38202	0.21348	0.01124	0.0	0.67486	0.92509	0.78041
	USAD [11]	0.22472	0.20225	0.30337	0.08989	0.34831	0.0	0.77816	0.85393	0.81429
	Ours	0.12360	0.15730	0.14607	0.19101	0.02247	0.0	0.86713	**0.92884**	**0.89693**
II Location Service	OneClassSVM[67]	0.14475	0.13226	0.01613	0.0	0.84194	0.0	0.88609	0.71935	0.79407
	iForest [48]	0.03619	0.03226	0.02258	0.5	0.08925	0.0	0.96513	0.80358	0.87698
	Mazzawi et al. [52]	0.00844	0.01505	0.02043	0.44086	0.99247	0.55914	0.95223	0.33584	0.49656
	DeepLog [21]	0.34861	0.75591	0.69677	0.0	0.16022	0.0	0.61691	0.94659	0.74699
	USAD [11]	0.18938	0.26667	0.17097	0.0	0.34839	0.0	0.81386	0.88387	0.84742
	Ours	0.04222	0.03871	0.03118	0.0	0.00430	0.0	**0.96535**	**0.99857**	**0.98168**



(a) Attention weights between operations.

T	Key	Statement
t1	321	INSERT INTO t_cell_fp_9(pnci, gridId, fps) VALUES ($1, $2, $3)
t2	358	SELECT * FROM t_cell_fp_9 WHERE pnci=$1 and gridId IN ($2, ... $36)
t3	31	INSERT INTO t_cell_fp_3(pnci, gridId, fps) VALUES ($1, $2, $3)
t4	230	SELECT * FROM t_cell_fp_3 WHERE pnci=$1 and gridId IN ($2, ... $13)
t5	460	SELECT * FROM t_cell_fp_3 WHERE pnci=$1 and gridId IN ($2, ... $9)
t6	128	INSERT INTO t_cell_fp_9(pnci, gridId, fps) VALUES ($1, $2, $3) ...($31, $32, $33)
t7	150	SELECT * FROM t_cell_fp_3 WHERE pnci=$1 and gridId IN ($2, ... $16)
t8	236	SELECT * FROM t_cell_fp_3 WHERE pnci=$1 and gridId IN ($2, $3)
t9	584	INSERT INTO t_cell_fp_3 (pnci, gridId, fps) VALUES ($1, $2, $3), ($4, $5, $6)
t10	45	INSERT INTO t_cell_picn_3(pnci, pi, cn) VALUES ($1, $2, $3)
t11	304	SELECT * FROM t_cell_fp_3 WHERE pnci=$1 and gridId IN ($2, $3)
t12	251	INSERT INTO t_cell_fp_3 (pnci, gridId, fps) VALUES ($1, $2, $3) ... ($64, $65, $66)
t13	45	INSERT INTO t_cell_picn_3(pnci, pi, cn) VALUES ($1, $2, $3)

(b) The keys and corresponding SQL statements of our normal session example.

Figure 6: The attention weights visualization for a normal session example. The red rectangle in each row highlights the most relevant contextual operation in inferring the semantics of one individual operation. The semantic relevance of operations in the red rectangle are also reflected in their SQL statements.

this is a normal session example where the key sequence is [321, 358, 31, 230, 460, 128, 150, 236, 584, 45, 304, 251, 45] and their corresponding SQL statements are listed in Figure 6(b). Each row in Figure 6(a) shows the attention weights of an individual operation in y-axis to its operation contexts in x-axis. Meanwhile, the red square highlights the most relevant (*i.e.,* highest weight) context in inferring the semantics of one individual operation. We can observe that operation 358 and 128 are two consecutive operations on table t_cell_fp_9. It means that the semantics of operation 128 is most relevant to operation 358, demonstrated by the red square in cell (X:358, Y:128). Besides, the similar queries on table t_cell_fp_3 (*i.e.,* operations 460, 150 and 236) indicate that their semantics are highly relevant. Trans-DAS captures such semantic patterns, *i.e.,* the two red squares in (X:150, Y:460) and (X:150, Y:236). Thus, Trans-DAS is effective to learn semantic information from operations contexts.

6.3 Analysis of Trans-DAS

In this section, we evaluate the impact of our designs in Trans-DAS to the anomaly detection performance of UCAD. Compared with the original Transformer model, Trans-DAS has three designs including an embedding layer without operation order information, a masking mechanism utilizing bidirectional contexts, and a new training objective utilizing triplet loss. Specifically, we use the original Transformer as the base model, and then add each design separately to generate a new variant. Table 3 presents the evaluation results of the original Transformer and these variants. We omit $\mathcal{A}3$ as the FNRs of all variants are zero.

(0) *Base Transformer.* The base model is Transformer with learnable position embedding and its masking mechanism in the decoder masks all the future operations. Besides, we use the one-class cross-entropy as its loss function. We observe that the base model has the lowest F1-score in both scenarios. In addition, its FPRs are higher than 0.2 in Scenario-I and close to 0.1 in Scenario-II, which are much worse than other variants.

(1) *Our embedding layer.* Compared with the base model, our variant improves the performance in Scenario-I (*i.e.,* the F1-score rises from 0.86731 to 0.87434 and the precision increased by 0.01575), yet slightly impairs the performance in Scenario-II. The reason behind this result is that the average session length of Scenario-II is much longer than that of Scenario-I. Although removing the order information increases its applicability to heterogeneous user access patterns, it also reduces the number of learnable parameters and then weakens its capability in learning the complex semantics of long sessions. Fortunately, this issue can be addressed by integrating our following two designs.

(2) *Our masking mechanism.* We observe that embedding bidirectional contexts is also effective to improve the detection performance, *i.e.,* the F1-score rises from 0.86731 to 0.88471 and from 0.95721 to 0.96991 in two scenarios, respectively. Both the FPR and

Table 3: The contribution of our new designs in Trans-DAS. Up arrows mean performance increases to the Base Transformer.

Scenario	Model Variants	FPR			FNR		P	R	F1
		$\mathcal{V}1$	$\mathcal{V}2$	$\mathcal{V}3$	$\mathcal{A}1$	$\mathcal{A}2$			
I Commenting Application	Base Transformer	0.20225	0.22472	0.21348	0.19101	0.02247	0.81311	0.92884	0.86713
	Our embedding layer	0.17977	0.19101	0.20225	0.20225	0.02247	0.82886 ↑	0.92509	0.87434 ↑
	Our masking mechanism	0.16265	0.16725	0.15791	0.19101	0.02247	0.84360 ↑	0.92884	0.88417 ↑
	Our training objective	0.15730	0.11236	0.13483	0.22472	0.02247	0.87189 ↑	0.91760	0.89416 ↑
	Trans-DAS	0.12360	0.15730	0.14607	0.19101	0.02247	0.86713 ↑	0.92884	**0.89693** ↑
II Location Service	Base Transformer	0.09530	0.09140	0.08602	0.0	0.00538	0.91945	0.99821	0.95721
	Our embedding layer	0.09771	0.09677	0.09570	0.0	0.00538	0.91461	0.99821	0.95458
	Our masking mechanism	0.06996	0.06452	0.05699	0.0	0.00215	0.94221 ↑	0.99928 ↑	0.96991 ↑
	Our training objective	0.03257	0.03763	0.03763	0.04624	0.03333	0.96550 ↑	0.97312	0.96930 ↑
	Trans-DAS	0.03860	0.04194	0.03226	0.0	0.00322	0.96535 ↑	0.99857 ↑	**0.98168** ↑

FNR have decreased by a certain degree. Consequently, the bidirectional contexts does help to accurately capture the contextual intent and contribute to detecting more stealthy anomalies.

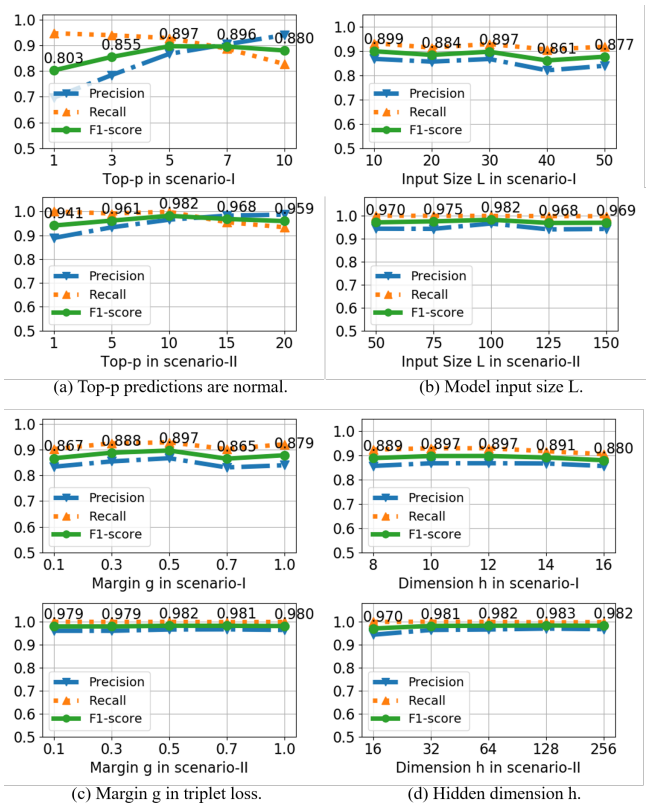
(3) *Our training objective.* We find that this variant improves the F1-score by about 0.02 on average. In particular, the precision is increased by more than 0.05 and the FPR is declined by 0.07865 and 0.05496 in two scenarios, respectively. This performance improvement indicates that our training objective is useful for enlarging the difference among the embedded semantics and then improving the detection performance.

Finally, by integrating these designs in the anomaly detection module, UCAD achieves the highest F1-score on both scenarios and the improvement of precision is around 0.05 on average. Meanwhile, with the same or even lower FNRs to base model, UCAD decreases the FPR by non-trivial margins, *i.e.,* up to about 62.5% in Scenarios-I and 38.9% in Scenario-II. Thus, our designs allow UCAD to effectively learn the semantic information and contextual intent, to perform an accurate data access anomaly detection.

6.4 Parameters Evaluation

Parameter Sensitivity. We evaluate the impacts of four major hyper-parameters on the performance of UCAD: (1) Top-p in online detection. (2) The input size L of Trans-DAS. (3) The margin g in our training objective. (4) The latent dimension h of the embedding layer. When testing one parameter, the others are fixed to the default values as listed in Section 6.1.

Figure 7(a) shows the influence of p on detection performance. Since the top-p prediction results mean operations with normal intent, a larger p value naturally causes a higher precision yet a lower recall. When varying p from 1 to 10 in Scenario-I, the F1-score rises from 0.803 to a peak of 0.897 (at p=5) and then falls slightly. We observe a similar tendency in Scenario-II and the optimal F1-score is 0.982 (at p=10) while the worst F1-score is 0.941 (at p=1). Figure 7(b) depicts the effect of different L. A larger L means more operation contexts can be used for semantics understanding, yet their semantics will be more complex and harder to analyze. Therefore, setting L to the average length of sessions (*i.e.,* about 30 in Scenario-I and 100 in Scenario-II) can achieve a better performance. Figure 7(c) presents the detection performance under different g. This parameter controls the relative distance between an anchor to positive and negative instances. It can be observed that UCAD is



(a) Top-p predictions are normal. (b) Model input size L.

(c) Margin g in triplet loss. (d) Hidden dimension h.

Figure 7: The impact of major hyper-parameters (p, L, g, h) on detection performance. The results indicate that Trans-DAS is nonsensitive to hyper-parameters.

nonsensitive to g and the change of F1-score is within 0.04. Lastly, we display the influence of h in Figure 7(d). The F1-score is relatively stable and shows a minor change of only 0.017 and 0.013 in two scenarios, respectively.

Overall, the detection performance of UCAD is nonsensitive to hyper-parameters and the variation of F1-score is less than 0.04 in most cases, indicating that the good capability of UCAD is founded on our elaborate designs rather than hyper-parameters tuning.

Table 4: The performance and training time (per epoch) under different latent dimension h in Scenario-II.

Dimension h	16	32	64	128	256
Time (s)	41	43	49	62	83
F1-score	0.96989	0.98099	0.98168	0.98268	0.98183

Table 5: The performance and training time (per epoch) under different input size L in Scenario-II.

Input size L	50	75	100	125	150
Time (s)	16	30	49	74	105
F1-score	0.97025	0.97473	0.98168	0.96783	0.96866

Training Time. Another issue worth concerning is the training time of UCAD (mainly the Trans-DAS), since an administrator may perform periodical offline training to address the problem of concept drift. Thus, a smaller training time is more desirable. In particular, we evaluate the training time of UCAD[7] under different L and h values in Scenario-II, which contains more training sessions, while omitting p and g as they are independent to the training time. Table 4 and 5 show the average training time for each epoch and the corresponding F1-score under different parameter values. We can see that the training time increases linearly with both L and h, while the F1-score changes slightly. In general, a smaller L or h can save a lot of training time and retain good detection capabilities.

Therefore, an administrator can choose either smaller values for faster training at the sacrifice of a little decline in the F1-score, or moderate parameter values to pursue a better detection performance. For instance, when L is 50, UCAD only requires 16 seconds to perform a training epoch and the F1-score is 0.97025, the training time is reduced by 66.7% at a cost of only 0.011 in F1-score.

6.5 Robustness

In this section, we investigate the robustness of UCAD to abnormal data, *i.e.,* the detection performance when the supposed normal training set for Trans-DAS actually contains abnormal data (which we refer to as hybrid dataset). To generate the hybrid dataset, we randomly choose synthetic abnormal sessions and insert them into our training set at different ratios. The detection performance is shown in Figure 8(a) and 8(b). We can see that the overall performance slowly declines as the percentage of abnormal sessions increases in both scenarios. In particular, when the anomaly percentage increases to 20%, the reduction of F1-score in Scenario-I is about 0.13 while that in Scenario-II is about 0.08. UCAD performs better in Scenario-II since this scenario contains much more training sessions than Scenario-I, so that the Trans-DAS can learn more normal semantic information to against the interfere of abnormal training data. When the anomaly percentage reaches up to 20%, UCAD still keeps an F1-score of nearly 0.9 in Scenario-II, which is still effective to detect the data access anomalies accurately.

We also show the influence of abnormal training data on the baseline methods in Figure 8(c) and 8(d). In particular, since OneClassSVM and iForest algorithms have specific hyper-parameters identifying the proportion of contamination in the training data, we adjust these hyper-parameters multiple times and report their best result. The method proposed in [52] performs poorly in the presence of any anomaly percentage. Meanwhile, the performance of other two methods (*i.e.,* DeepLog and USAD) also shows certain degree of reduction (*i.e.,* 0.1 and 0.09 on average). Compared with baselines, UCAD can still achieve the best performance in most cases, which demonstrates that it is robust to abnormal training data and more applicable to different real-world scenarios.

[7]Our evaluation platform is with Intel Core i7-8700 CPU, 32GB RAM and no GPU.

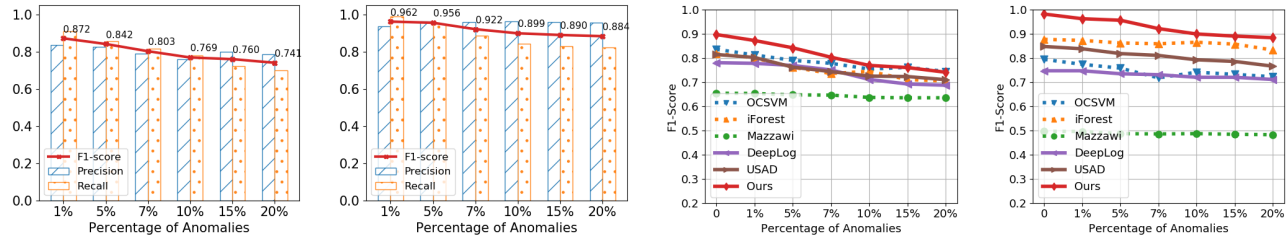
6.6 Transferability

We further investigate the transferability of UCAD (especially the Trans-DAS) to a relevant task: system log anomaly detection. The evaluation is performed on three public system log datasets: (1) The **HDFS** [84] dataset, which is generated by running Hadoop-based map-reduce tasks on more than 200 Amazon EC2 nodes. It contains 11,197,954 log entries and 2.9% of them are abnormal. The log entries are grouped into sessions by the block_id field. (2) The **BGL** [59] dataset, which is generated by the Blue Gene/L supercomputer. It contains 4,747,963 log entries and 348,460 of them are abnormal. (3) The **Thunderbird** [59] dataset, which is collected from a supercomputer system and contains 211,212,192 log messages of which 3,248,239 are abnormal. Since there are no session identifiers in BGL and Thunderbird datasets, we use the preprocessing method applied in DeepLog [21] to generate the session sequences for evaluation.

The parameters of Trans-DAS are set as: L=10, g=0.5, h=64. We compare Trans-DAS with two representative system log anomaly detection methods, i.e., LogCluster [46] and DeepLog [21]. Table 6 illustrates that UCAD achieves the highest recall (*i.e.,* 0.97213, 0.95823 and 1.0 in three datasets, respectively) and the highest F1-score of 0.93063 and 0.94225 in BGL and Thunderbird datasets, respectively. Only the precision is slightly lower than that of LogCluster. The reason behind is that the behavioral patterns of an application are typically less diverse than those of human beings. Thus, the normal application sessions exhibit relatively fixed patterns that can be learned according to the order of entries. Yet UCAD is specifically designed and optimized towards handling the challenge of heterogeneous access patterns. The generated false alarms result in a slightly lower precision in this task. However, we argue that a higher recall is much more critical for system anomaly detection in production, since it is much less expensive to pay a little extra engineering toll to handle those false alerts than missing a malicious operation.

6.7 User Study

In this section, we discuss two real cases where UCAD helps database administrators (DBAs) to locate data access anomalies in two real-world production systems. As shown in Figure 9(a), the first case is regarding living video commentary, where a bot impersonates the legitimate client to post *danmu* (*i.e.,* bullet screen, a form of video commentary). The DBAs locate a suspicious session (9->23->11->4->3...), where the operations 11->4 are identified as anomalies by UCAD. The DBAs utilize their domain knowledge and first study the normal session in Figure 9(a) which indicates the following actions: (1) the danmus were inserted into table danmu_display in a fixed frequency (operation 1); (2) a user clicked the “open danmu" button when watching videos, so that the danmus were selected



(a) Impact of abnormal training data on Trans-DAS in Scenario-I.

(b) Impact of abnormal training data on Trans-DAS in Scenario-II.

(c) Impact of abnormal training data on all methods in Scenario-I.

(d) Impact of abnormal training data on all methods in Scenario-II.

Figure 8: The impact of abnormal sessions in the training set. (a) (b) show that the overall performance of Trans-DAS slowly declines as the percentage of abnormal data increases. (c) (d) show the impact of abnormal data on all approaches and Trans-DAS remains the highest F1-score in most cases.

Table 6: The detection performance on HDFS, BGL and Thunderbird datasets. It demonstrates that UCAD can be effectively transferred to relevant tasks.

Dataset	Methods	LogCluster	DeepLog	Ours
HDFS	Precision	**0.87371**	0.87022	0.84248
	Recall	0.74109	0.96073	**0.97213**
	F1-score	0.80195	**0.91324**	0.90267
BGL	Precision	**0.95463**	0.89741	0.90449
	Recall	0.64012	0.82783	**0.95823**
	F1-score	0.76636	0.86122	**0.93063**
Thunder	Precision	**0.98280**	0.77421	0.89080
	Recall	0.42782	1.00000	**1.00000**
	F1-score	0.59614	0.87273	**0.94225**

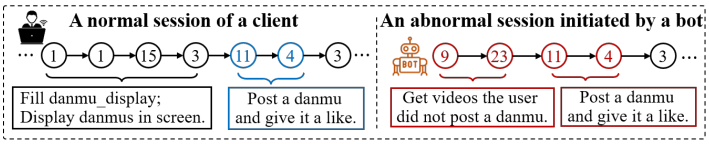
and displayed on the screen (operations 15->3); (3) the user posted a danmu and gave it a like. However, the suspicious session exhibits different actions. By analyzing its concrete SQL statements along with auxiliary messages (*e.g.*, the timestamp and user identity), the DBAs notice that the user first obtained videos on which she did not post any danmu (operations 9->23), and then immediately posted one and gave it a like (operations 11->4). The user did not click the "open" button (meaning neither danmus nor the danmu input box was available), yet she still posted one and gave a like to an *invisible* danmu. In addition, the DBAs track this suspicious user for several days and notice the same abnormal operation sequence performed every day at 12:00 noon. Thus, with high confidence, the DBAs confirm that the "user" is a bot designed to finish daily task to get rewards (*e.g.*, virtual coins or experience points).

As shown in Figure 9(b), the second anomaly case is related with the location service, where a maliciously repackaged app steals the credential of a normal app on the same device and reports manipulated location data. The location service is responsible for obtaining the locations of mobile devices (in form of the latitudes and longitudes) that install our mobile apps. All our authenticated apps are able to report location data. The actual authentication process is complicated. An extremely simplified view (related with database) is that a combination of operations 61 and 512 is necessary for access verification. Afterwards, the app can issue other operations like reading data from table loc_rm (operation 264) and inserting data into table loc_rmf for offline access (operation 300), as shown in the normal session of Figure 9(b). In a suspicious session reported by UCAD, it detects consecutive insert operations (73 in red) to table loc_rm, indicating very frequent updates of locations during a short period of time. A deeper analysis by the DBAs reveals a critical vulnerability rooted in access control. In particular, a maliciously repackaged app managed to steal the authentication credential of a normal app on the same device and started to report manipulated location data. This is an example that anomalies in database operations reveal critical vulnerabilities that could otherwise manifest in a larger scale and result in catastrophic data losses (for instance, what if the attacker is able to access data other than locations?).

7 DISCUSSION

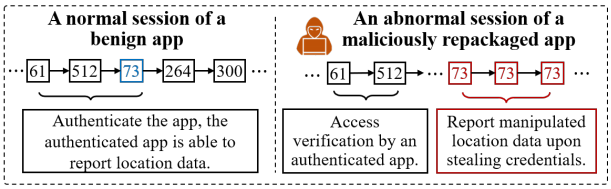
Adversarial Examples and Data Poisoning. Given that Trans-DAS is a deep learning model, attackers may use adversarial example [28, 88] and data poisoning [37] to evade detection. We argue that both methods are less practical in database systems. For safety, a typical database system does not allow users to craft their own queries. It means that a successful adversarial example can only use limited statement templates to achieve its malicious goal while disguising as seemingly normal intent, otherwise it triggers alert like statement parse warning. Data poisoning requires a certain number of poisoned instances to be injected into training set before model training. Fortunately, note that the next round of offline training uses the normal sessions verified by our previously trained system, it will be difficult for attackers to bypass our detection and inject enough poisoned data into the training set. Besides, the robustness of Trans-DAS (as evaluated in Section 6.5) has shown that Trans-DAS is not easily affected by a small amount of poisoned training data. Poisoning defenses [60, 72] can be utilized to improve the robustness. We leave it as an interesting future work.

Lifelong Learning. In order to learn the latest normal semantic information, our system currently adopts the fine-tuning strategy, which collects the new normal sessions verified by our system and fine-tune the previously trained anomaly detection module. Despite its effectiveness in handling the problem of concept drift, it may have another minor shortcoming that the model fine-tuned on new data may generate false alarms on forgotten normal patterns [23]. A future direction for solving this issue is to apply lifelong learning [17], which utilizes regularization techniques [19, 45], memory



Key	Statement
1	insert into danmu_display(videoKey, danmuId,...) values($1,...);
3	update danmu_content_clips set danmuCount=$1 where videoKey=$2;
4	updata danmu_online set likeCount=$1 where videoKey=$2 and danmuId=$3;
9	select videoKey,... from danmu_offline where encUserId=$1;
11	insert into danmu_online(danmuId, videoKey, appId...) values($1,...);
15	select videoKey,... from danmu_display where videoKey=$1;
23	select videoKey,... from danmu_content_clips where videoKey not in ($1,...);

(a) An anomaly case of a bot in the online commenting scenario.



Key	Statement
61	insert into loc_buffer(mac, gridId, lat, lon, rssi) values ($1, $2, $3, $4, $5);
73	insert into loc_rm(mac, gridId, lat, lon, rssi) values ($1, $2, $3, $4, $5);
264	select * from loc_rm where mac=$1;
300	insert into loc_rmf(mac, blockId, lat, lon, rssi) values ($1, $2, $3, $4, $5);
512	select * from loc_buffer where mac=$1;

(b) An anomaly case of a vulnerability in the location service scenario.

Figure 9: Two real-world anomaly cases reported by UCAD in our production systems. The first case is caused by a bot designed to finish daily task to get rewards, and the second case is due to a critical vulnerability rooted in access control.

replay [8, 62] or model adaptation [36] to make learning-based models adapt to new data without forgetting its existing knowledge.

Limitations. Our study has two potential limitations. First, our system assumes that the operation log is reliable and intact, which may be tampered by sophisticated attackers. Meanwhile, its detection result may also be manipulated by the hidden malware. In the future work, we plan to deploy the log recorder and our system in trusted execution environments like Intel SGX [10, 32, 53] to defend against these threats. Second, we observe that false positives occur frequently in specific production environments and result in more expert efforts. We suspect the reason is that the generalization ability on some rarely appeared normal patterns is insufficient. To address this limitation, a data augmentation process [74] can be incorporated into our system to enhance the detection precision.

8 RELATED WORK

Database Anomaly Detection. A series of approaches have been proposed for database anomaly detection. The syntax-based methods [14, 39, 40, 64] focus on analyzing the syntax of SQL statements. Specifically, in order to detect abnormal data access operations, Sallam et al. [64] took role-based syntax analysis and Hussain et al. [39] built behavior profiles representing normal executing syntax. Fadolalkarim et al. [22] captured operation syntax via dynamical analysis to detect data leakages. The data-based methods [43, 51] extract the statistics related to the queried data. The context-based methods [7, 25, 42, 83] use the information relevant to operation execution or user attributes. Besides, the hybrid methods [52, 65, 66, 71] integrate the design primitives of other approaches to detect more anomalies. For example, Mazzawi et al. [52] used both SQL syntax and behavioral attributes to profile the data access operations. Compared with them, our work focuses on capturing the semantics of individual operations and the contextual intent of operations, such that it can detect the stealthy data access anomalies

Unsupervised Anomaly Detection. Unsupervised anomaly detection methods have been widely used in many scenarios including clustering-based [31, 50], nearest neighbor-based [13, 29], isolation-based [48], and kernel-based [9, 85] detection. Recently, the deep learning based approaches become more popular. Specifically, DeepLog [21] uses LSTM to detect online abnormal operations in system logs. LogAnomaly [54] uses dLCE [58] to embed log words into templates for temporal dependence analysis. The Deep Autoencoding Gaussian Mixture Model [87] estimates the density distribution of multidimensional data for efficient anomaly detection. The OmniAnomaly [73] learns robust representations for multivariate time series data via stochastic variable connection and planar normalizing flows. The USAD [11] is an unsupervised anomaly detection method based on adversarially trained Autoencoders. In this work, we utilize the self-attention mechanism to build our Trans-DAS model and develop several new designs to effectively learn the semantic information and the contextual intent of database access operations for database anomaly detection.

Applications of Attention Mechanism. The attention mechanism has shown great power in various areas including NLP [12, 18, 69, 79], CV [82, 86], and recommendation systems [30, 41]. Some recent proposals [57, 63] learn semantic embeddings via the self-attention mechanism to detect point anomalies. In our work, we propose new designs built on the self-attention mechanism to accurately detect the stealthy abnormal data access operations.

9 CONCLUSION

In this paper, we propose a novel unsupervised anomaly detection system, UCAD, for database. UCAD detects data access operations based on the semantic analysis of operations. In particular, we develop Trans-DAS model for UCAD to accurately learn the semantics of individual operations and the intent of the operation sequence. Trans-DAS is built on the attention mechanism that captures the relevance between any pair of operations in the sequence. The extensive evaluations in two real-world typical scenarios demonstrate the effectiveness of UCAD. The results illustrate that UCAD can achieve the F1-scores of 0.89693 and 0.98168 in two scenarios, respectively. The comparison experiments between UCAD and baseline methods show that UCAD has obvious performance advantage. Specifically, UCAD is robust to detect abnormal data access operations for different tasks under various settings.

ACKNOWLEDGMENTS

This work is supported in part by the National Key R&D Project of China under Grant 2021ZD0110502, NSFC under Grant 62132011 and 61925205, Huawei, TAL education, and BNRist. Guoliang Li and Qi Li are the corresponding authors of this paper.

REFERENCES

[1] 2013. Snowden—Ultimate Insider Threat Missed by NSA Security. https://freebeacon.com/national-security/cyber-threat-snowden-insider-threat-at-nsa/.

[2] 2020. 17 Real Examples of Insider Threats. https://www.tessian.com/blog/insider-threats-types-and-real-world-examples/.

[3] 2020. Cost of a Data Breach Report. https://www.ibm.com/security/data-breach.

[4] 2021. Insider Threats Statistics. https://techjury.net/blog/insider-threat-statistics/.

[5] 2021. Masters of Mimicry. https://www.ptsecurity.com/ww-en/analytics/pt-esc-threat-intelligence/new-apt-group-chamelgang/.

[6] Muhammad Umar Aftab, Muhammad Asif Habib, Nasir Mehmood, Mubeen Aslam, and Muhammad Irfan. 2015. Attributed role based access control model. In *Proceedings of 2015 Conference on Information Assurance and Cyber Security (CIACS)*. IEEE, 83–89.

[7] Mahdi Alizadeh, Sander Peters, Sandro Etalle, and Nicola Zannone. 2018. Behavior analysis in the medical sector: theory and practice. In *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*. 1637–1646.

[8] Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. 2019. Gradient based sample selection for online continual learning. *arXiv preprint arXiv:1903.08671* (2019).

[9] Mennatallah Amer, Markus Goldstein, and Slim Abdennadher. 2013. Enhancing one-class support vector machines for unsupervised anomaly detection. In *Proceedings of ACM SIGKDD Workshop on Outlier Detection and Description*. 8–15.

[10] Ittai Anati, Shay Gueron, Simon Johnson, and Vincent Scarlata. 2013. Innovative technology for CPU based attestation and sealing. In *Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy*, Vol. 13. ACM New York, NY, USA, 7.

[11] Julien Audibert, Pietro Michiardi, Frédéric Guyard, Sébastien Marti, and Maria A Zuluaga. 2020. USAD: UnSupervised Anomaly Detection on Multivariate Time Series. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 3395–3404.

[12] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of International Conference on Learning Representations*.

[13] Liron Bergman, Niv Cohen, and Yedid Hoshen. 2020. Deep nearest neighbor anomaly detection. *arXiv preprint arXiv:2002.10445* (2020).

[14] Elisa Bertino, Evimaria Terzi, Ashish Kamra, and Athena Vakali. 2005. Intrusion detection in RBAC-administered databases. In *Proceedings of 21st Annual Computer Security Applications Conference (ACSAC'05)*. IEEE, 10–pp.

[15] Sandeep Bhatkar, Daniel C DuVarney, and Ron Sekar. 2003. Address Obfuscation: An Efficient Approach to Combat a Broad Range of Memory Error Exploits.. In *Proceedings of USENIX Security Symposium*, Vol. 12. 291–301.

[16] Andrea Bittau, Adam Belay, Ali Mashtizadeh, David Mazieres, and Dan Boneh. 2014. Hacking blind. In *Proceedings of 2014 IEEE Symposium on Security and Privacy*. IEEE, 227–242.

[17] Zhiyuan Chen and Bing Liu. 2018. Lifelong machine learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 12, 3 (2018), 1–207.

[18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[19] Min Du, Zhi Chen, Chang Liu, Rajvardhan Oak, and Dawn Song. 2019. Lifelong Anomaly Detection Through Unlearning. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 1283–1297.

[20] Min Du and Feifei Li. 2016. Spell: Streaming parsing of system event logs. In *Proceedings of 2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, 859–864.

[21] Min Du, Feifei Li, Guineng Zheng, and Vivek Srikumar. 2017. Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 1285–1298.

[22] Daren Fadolalkarim, Elisa Bertino, and Asmaa Sallam. 2020. An Anomaly Detection System for the Protection of Relational Database Systems against Data Leakage by Application Programs. In *Proceedings of 2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 265–276.

[23] Robert M French. 1999. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences* 3, 4 (1999), 128–135.

[24] Qiang Fu, Jian-Guang Lou, Yi Wang, and Jiang Li. 2009. Execution anomaly detection in distributed systems through unstructured log analysis. In *Proceedings of 2009 IEEE International Conference on Data Mining (ICDM)*. IEEE, 149–158.

[25] Ma'ayan Gafny, Asaf Shabtai, Lior Rokach, and Yuval Elovici. 2011. Poster: Applying Unsupervised Context-Based Analysis for Detecting Unauthorized Data Disclosure. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*.

[26] Junhao Gan and Yufei Tao. 2015. DBSCAN revisited: mis-claim, un-fixability, and approximation. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. 519–530.

[27] Yoav Goldberg and Omer Levy. 2014. word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722* (2014).

[28] Kathrin Grosse, Nicolas Papernot, Praveen Manoharan, Michael Backes, and Patrick McDaniel. 2017. Adversarial examples for malware detection. In *Proceedings of European Symposium on Research in Computer Security*. Springer, 62–79.

[29] Xiaoyi Gu, Leman Akoglu, and Alessandro Rinaldo. 2019. Statistical analysis of nearest neighbor methods for anomaly detection. In *Advances in Neural Information Processing Systems*. 10923–10933.

[30] Yulong Gu, Zhuoye Ding, Shuaiqiang Wang, Lixin Zou, Yiding Liu, and Dawei Yin. 2020. Deep Multifaceted Transformers for Multi-objective Ranking in Large-Scale E-commerce Recommender Systems. In *Proceedings of ACM Conference on Information and Knowledge Management*. 2493–2500.

[31] Zengyou He, Xiaofei Xu, and Shengchun Deng. 2003. Discovering cluster-based local outliers. *Pattern Recognition Letters* 24, 9-10 (2003), 1641–1650.

[32] Matthew Hoekstra, Reshma Lal, Pradeep Pappachan, Vinay Phegade, and Juan Del Cuvillo. 2013. Using innovative instructions to create trustworthy software solutions. *Hardware and Architectural Support for Security and Privacy (HASP)* 11, 10.1145 (2013), 2487726–2488370.

[33] Vincent C Hu, David Ferraiolo, Rick Kuhn, Arthur R Friedman, Alan J Lang, Margaret M Cogdell, Adam Schnitzer, Kenneth Sandlin, Robert Miller, Karen Scarfone, et al. 2013. Guide to attribute based access control (ABAC) definition and considerations (draft). *NIST Special Publication* 800, 162 (2013).

[34] Vincent C. Hu, D. Richard Kuhn, David F. Ferraiolo, and Jeffrey Voas. 2015. Attribute-Based Access Control. *Computer* 48, 2 (2015), 85–88. https://doi.org/10.1109/MC.2015.33

[35] Vincent C Hu, D Richard Kuhn, David F Ferraiolo, and Jeffrey Voas. 2015. Attribute-based access control. *Computer* 48, 2 (2015), 85–88.

[36] Wenpeng Hu, Zhou Lin, Bing Liu, Chongyang Tao, Zhengwei Tao, Jinwen Ma, Dongyan Zhao, and Rui Yan. 2018. Overcoming catastrophic forgetting for continual learning via model adaptation. In *Proceedings of International Conference on Learning Representations (ICLR)*.

[37] Hai Huang, Jiaming Mu, Neil Zhenqiang Gong, Qi Li, Bin Liu, and Mingwei Xu. 2021. Data Poisoning Attacks to Deep Learning Based Recommender Systems. *arXiv preprint arXiv:2101.02644* (2021).

[38] Junbeom Hur and Dong Kun Noh. 2010. Attribute-based access control with efficient revocation in data outsourcing systems. *IEEE Transactions on Parallel and Distributed Systems* 22, 7 (2010), 1214–1221.

[39] Syed Rafiul Hussain, Asmaa M Sallam, and Elisa Bertino. 2015. Detanom: Detecting anomalous database transactions by insiders. In *Proceedings of the 5th ACM Conference on Data and Application Security and Privacy*. 25–35.

[40] Ashish Kamra, Evimaria Terzi, and Elisa Bertino. 2008. Detecting anomalous access patterns in relational databases. *The International Journal on Very Large Data Bases* 17, 5 (2008), 1063–1077.

[41] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *Proceedings of 2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 197–206.

[42] Muhammad Imran Khan and Simon N Foley. 2016. Detecting anomalous behavior in DBMS logs. In *Proceedings of International Conference on Risks and Security of Internet and Systems*. Springer, 147–152.

[43] Muhammad Imran Khan, Barry O'Sullivan, and Simon N Foley. 2017. A semantic approach to frequency based anomaly detection of insider access in database management systems. In *Proceedings of International Conference on Risks and Security of Internet and Systems*. Springer, 18–28.

[44] Grzegorz Kondrak. 2005. N-gram similarity and distance. In *Proceedings of International Symposium on String Processing and Information Retrieval*. Springer, 115–126.

[45] Zhizhong Li and Derek Hoiem. 2017. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40, 12 (2017), 2935–2947.

[46] Qingwei Lin, Hongyu Zhang, Jian-Guang Lou, Yu Zhang, and Xuewei Chen. 2016. Log clustering based problem identification for online service systems. In *Proceedings of 2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE)*. IEEE, 102–111.

[47] Fucheng Liu, Yu Wen, Dongxue Zhang, Xihe Jiang, Xinyu Xing, and Dan Meng. 2019. Log2vec: a heterogeneous graph embedding based approach for detecting cyber threats within enterprise. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 1777–1794.

[48] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2008. Isolation forest. In *Proceedings of 2008 Eighth IEEE International Conference on Data Mining*. IEEE, 413–422.

[49] Jie Lu, Anjin Liu, Fan Dong, Feng Gu, Joao Gama, and Guangquan Zhang. 2018. Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering* (2018), 2346–2363.

[50] Alejandro Marcos Alvarez, Makoto Yamada, Akisato Kimura, and Tomoharu Iwata. 2013. Clustering-based anomaly detection in multi-view data. In *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management*. 1545–1548.

[51] Sunu Mathew, Michalis Petropoulos, Hung Q Ngo, and Shambhu Upadhyaya. 2010. A data-centric approach to insider attack detection in database systems. In *International Workshop on Recent Advances in Intrusion Detection (RAID)*. Springer, 382–401.

[52] Hanna Mazzawi, Gal Dalal, David Rozenblatz, Liat Ein-Dorx, Matan Niniox, and Ofer Lavi. 2017. Anomaly detection in large databases using behavioral patterning. In *Proceedings of 2017 IEEE 33rd International Conference on Data Engineering (ICDE)*. IEEE, 1140–1149.

[53] Frank McKeen, Ilya Alexandrovich, Alex Berenzon, Carlos V Rozas, Hisham Shafi, Vedvyas Shanbhogue, and Uday R Savagaonkar. 2013. Innovative instructions and software model for isolated execution. *Hardware and Architectural Support for Security and Privacy (HASP)* 10, 1 (2013).

[54] Weibin Meng, Ying Liu, Yichen Zhu, Shenglin Zhang, Dan Pei, Yuqing Liu, Yihao Chen, Ruizhi Zhang, Shimin Tao, Pei Sun, et al. 2019. LogAnomaly: Unsupervised Detection of Sequential and Quantitative Anomalies in Unstructured Logs. In *Proceedings of International Joint Conference on Artificial Intelligence*, Vol. 7. 4739–4745.

[55] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. arXiv:1301.3781 [cs.CL]

[56] Shinsuke Mori, Masafumi Nishimura, and Nobuyasu Itoh. 1998. Word clustering for a word bi-gram model. In *Proceedings of Fifth International Conference on Spoken Language Processing*.

[57] Sasho Nedelkoski, Jasmin Bogatinovski, Alexander Acker, Jorge Cardoso, and Odej Kao. 2020. Self-attentive classification-based anomaly detection in unstructured logs. *arXiv preprint arXiv:2008.09340* (2020).

[58] Kim Anh Nguyen, Sabine Schulte im Walde, and Ngoc Thang Vu. 2016. Integrating distributional lexical contrast into word embeddings for antonym-synonym distinction. *arXiv preprint arXiv:1605.07766* (2016).

[59] Adam Oliner and Jon Stearley. 2007. What Supercomputers Say: A Study of Five System Logs. In *Proceedings of 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*.

[60] Neehar Peri, Neal Gupta, W Ronny Huang, Liam Fowl, Chen Zhu, Soheil Feizi, Tom Goldstein, and John P Dickerson. [n.d.]. Deep k-nn defense against clean-label data poisoning attacks. In *Proceedings of European Conference on Computer Vision*.

[61] U. P. Rao and N. K. Singh. 2015. *Detection of Privilege Abuse in RBAC Administered Database*. Intelligent Systems in Science and Information.

[62] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy P Lillicrap, and Greg Wayne. 2018. Experience replay for continual learning. *arXiv preprint arXiv:1811.11682* (2018).

[63] Lukas Ruff, Yury Zemlyanskiy, Robert Vandermeulen, Thomas Schnake, and Marius Kloft. 2019. Self-attentive, multi-context one-class classification for unsupervised anomaly detection on text. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 4061–4071.

[64] Asmaa Sallam, Elisa Bertino, Syed Rafiul Hussain, David Landers, R Michael Lefler, and Donald Steiner. 2015. DBSAFE—an anomaly detection system to protect databases from exfiltration attempts. *IEEE Systems Journal* 11, 2 (2015), 483–493.

[65] Asmaa Sallam, Daren Fadolalkarim, Elisa Bertino, and Qian Xiao. 2016. Data and syntax centric anomaly detection for relational databases. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 6, 6 (2016), 231–239.

[66] Ricardo Jorge Santos, Jorge Bernardino, Marco Vieira, and Deolinda M. L. Rasteiro. 2012. Securing Data Warehouses from Web-Based Intrusions. In *Proceedings of Web Information Systems Engineering (WISE)*.

[67] Bernhard Schölkopf, John C Platt, John Shawe-Taylor, Alex J Smola, and Robert C Williamson. 2001. Estimating the support of a high-dimensional distribution. *Neural computation* 13, 7 (2001), 1443–1471.

[68] Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of Internaltional Conference on Computer Vision and Pattern Recogintion*. 815–823.

[69] Yelong Shen, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen. 2017. Reasonet: Learning to stop reading in machine comprehension. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1047–1055.

[70] Yun Shen, Enrico Mariconti, Pierre Antoine Vervier, and Gianluca Stringhini. 2018. Tiresias: Predicting security events through deep learning. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 592–605.

[71] A. Spalka and J. Lehnhardt. 2005. A comprehensive approach to anomaly detection in relational databases. In *Proceedings of Ifip Wg 113 Working Conference on Data & Applications Security*.

[72] Jacob Steinhardt, Pang Wei Koh, and Percy Liang. 2017. Certified defenses for data poisoning attacks. *arXiv preprint arXiv:1706.03691* (2017).

[73] Ya Su, Youjian Zhao, Chenhao Niu, Rong Liu, Wei Sun, and Dan Pei. 2019. Robust anomaly detection for multivariate time series through stochastic recurrent neural network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2828–2837.

[74] Lichao Sun, Congying Xia, Wenpeng Yin, Tingting Liang, Philip S Yu, and Lifang He. 2020. Mixup-Transfomer: Dynamic Data Augmentation for NLP Tasks. *arXiv preprint arXiv:2010.02394* (2020).

[75] Liang Tang, Tao Li, and Chang-Shing Perng. 2011. LogSig: Generating system events from raw textual logs. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*. 785–794.

[76] Andrija Tomović, Predrag Janičić, and Vlado Kešelj. 2006. n-Gram-based classification and unsupervised hierarchical clustering of genome sequences. *Computer Methods and Programs in Biomedicine* 81, 2 (2006), 137–153.

[77] Alexey Tsymbal. 2004. The problem of concept drift: definitions and related work. *Computer Science Department, Trinity College Dublin* (2004).

[78] Esko Ukkonen. 1992. Approximate string-matching with q-grams and maximal matches. *Theoretical Computer Science* 92, 1 (1992), 191–211.

[79] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*. 5998–6008.

[80] David Wagner and Paolo Soto. 2002. Mimicry attacks on host-based intrusion detection systems. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*. 255–264.

[81] Jin Wang, Yangning Tang, Shiming He, Changqing Zhao, Pradip Kumar Sharma, Osama Alfarraj, and Amr Tolba. 2020. LogEvent2vec: LogEvent-to-vector based anomaly detection for large-scale logs in internet of things. *Sensors* 20, 9 (2020), 2451.

[82] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. 2018. Cbam: Convolutional block attention module. In *Proceedings of Europeon Conference on Computer Vision*. 3–19.

[83] Garfield Zhiping Wu, Sylvia L Osborn, and Xin Jin. 2009. Database intrusion detection using role profiling with role hierarchy. In *Proceedings of Workshop on Secure Data Management*. Springer, 33–48.

[84] Wei Xu, Ling Huang, Armando Fox, David Patterson, and Michael Jordan. 2009. Online system problem detection by mining patterns of console logs. In *Proceedings of IEEE International Conference on Data Mining (ICDE)*. IEEE, 588–597.

[85] Ming Zhang, Boyi Xu, and Jie Gong. 2015. An anomaly detection model based on one-class svm to detect network intrusions. In *Proceedings of 2015 11th International Conference on Mobile Ad-hoc and Sensor Networks (MSN)*. IEEE, 102–107.

[86] Hengshuang Zhao, Jiaya Jia, and Vladlen Koltun. 2020. Exploring self-attention for image recognition. In *Proceedings of Internaltional Conference on Computer Vision and Pattern Recogintion*. 10076–10085.

[87] Bo Zong, Qi Song, Martin Renqiang Min, Wei Cheng, Cristian Lumezanu, Daeki Cho, and Haifeng Chen. 2018. Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In *Proceedings of International Conference on Learning Representations*.

[88] Wei Zou, Shujian Huang, Jun Xie, Xinyu Dai, and Jiajun Chen. 2019. A reinforced generation of adversarial examples for neural machine translation. *arXiv preprint arXiv:1911.03677* (2019).