Score-VAE: Root Cause Analysis for Federated-Learning-Based IoT Anomaly Detection

Jiamin Fan, Guoming Tang, *Member, IEEE*, Kui Wu, *Senior Member, IEEE*, Zhengan Zhao, *Graduate Student Member, IEEE*, Yang Zhou, and Shengqiang Huang

***Abstract*—Root cause analysis is the process of identifying the underlying factors responsible for triggering anomaly detection alarms. In the context of anomaly detection for Internet of Things (IoT) traffic, these alarms can be triggered by various factors, not all of which are malicious attacks. It is crucial to determine whether a malicious attack or benign operations cause an alarm. To address this challenge, we propose an innovative root cause analysis system called score-variational autoencoder (VAE), designed to complement existing IoT anomaly detection systems based on the federated learning (FL) framework. Score-VAE harnesses the full potential of the VAE network by integrating its training and testing schemes strategically. This integration enables Score-VAE to effectively utilize the generation and reconstruction capabilities of the VAE network. As a result, it exhibits excellent generalization, lifelong learning, collaboration, and privacy protection capabilities, all of which are essential for performing root cause analysis on IoT systems. We evaluate Score-VAE using real-world IoT trace data collected from various scenarios. The evaluation results demonstrate that Score-VAE accurately identifies the root causes behind alarms triggered by IoT anomaly detection systems. Furthermore, Score-VAE outperforms the baseline methods, providing superior performance in discovering root causes and delivering more accurate results.**

***Index Terms*—Internet of Things (IoT) traffic anomaly detection, machine learning (ML), root cause analysis.**

I. Introduction

A. Motivation

IN RECENT years, federated learning (FL) has been widely applied in the field of Internet of Things (IoT) anomaly detection [1], [2], [3], [4], [5], [6]. It employs data spread across various devices [4], [7], enabling the collaborative training of a robust global model while ensuring the privacy of local users. As shown in Fig. 1, a global model is trained for each type of IoT device since the same type of IoT devices may adopt similar configurations (e.g., default factory settings) and thus yield similar traffic patterns [4], [8].

Manuscript received 19 January 2023; revised 5 June 2023; accepted 14 June 2023. Date of publication 26 June 2023; date of current version 25 December 2023. This work was supported by Huawei Technologies Canada Company Ltd., through Huawei–UVic under Contract 6366. *(Corresponding author: Kui Wu.)*

Jiamin Fan, Kui Wu, and Zhengan Zhao are with the Department of Computer Science, University of Victoria, Victoria, BC V8P 4P1, Canada (e-mail: wkui@uvic.ca).

Guoming Tang is with the Network Communication Research Center, Peng Cheng Laboratory, Shenzhen 518055, China.

Yang Zhou and Shengqiang Huang are with the Vancouver Research Center, Huawei Technologies Canada Company Ltd., Vancouver, BC V5C 6S7, Canada.

Digital Object Identifier 10.1109/JIOT.2023.3289814

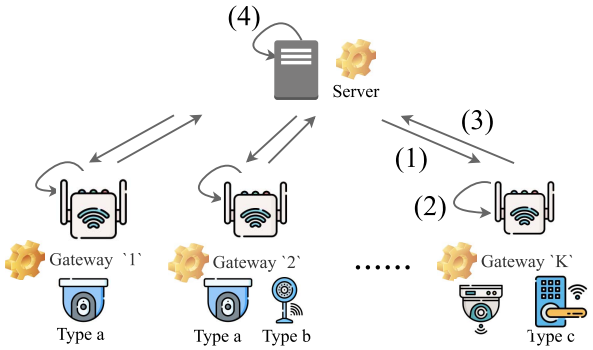


Fig. 1. Architecture of FL-based anomaly detection. The training process consists of four steps: 1) each client (i.e., gateway) downloads the initial global detection model; 2) the client updates the global model with local data; 3) the client sends the updated model to the server; and 4) the server aggregates the local models from the clients.

Nevertheless, our experimental observations in a real-world IoT system disclose that traffic from different clients might not be independent and identically distributed (IID),[1] resulting in a high volume of false positive (FP) events. A concrete example is illustrated in Fig. 2.

To tackle this issue, we employ root cause analysis. Root cause analysis has proven effective in identifying the core causes of anomalies in numerous fields. Nonetheless, when adopting them to analyze the anomalies detected by FL-based anomaly detection in IoT systems, we encounter the following problems.

1) Root cause analysis systems applied in IoT need stronger generalization capabilities. The volatile network environment within IoT systems results in significant fluctuations even for data originating from identical scenarios. Moreover, malicious attacks can manifest in novel and unprecedented forms. In such circumstances, it becomes crucial for the system to identify the most probable cause of the anomaly.
2) Collaborative capabilities are crucial for IoT systems. The diverse and complex nature of IoT anomaly scenarios poses a significant challenge for local users to independently train a robust root cause analysis model. However, the current root cause analysis solutions [11],

[1]The concept of IID assumes that the samples are independent of each other and identically distributed. The term non-IID refers to any settings beyond IIDnesses.

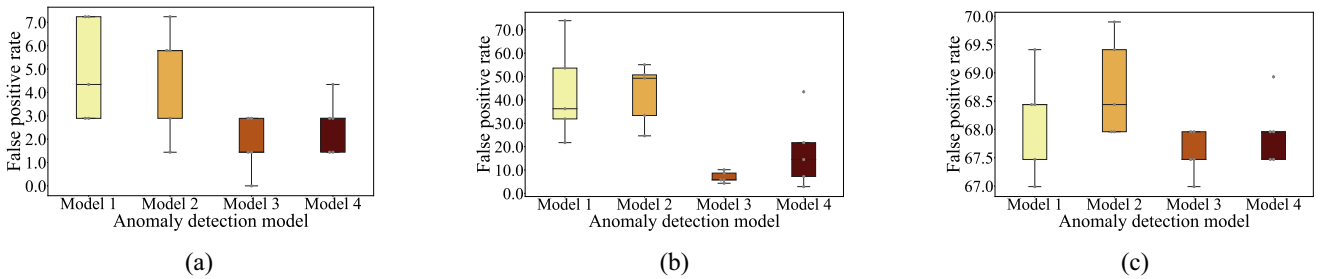


Fig. 2. Boxplot of FP rate (FPR, %) of different anomaly detection models in: (a) data from different clients are IID; (b) the training data and testing data from the same client are IID, but data from different clients are non-IID. This emulates the situation where different clients may have different network configurations or network conditions; (c) the training data from different clients are IID, but the training data and test data from the same client are non-IID. This happens when network conditions change significantly over time and such changes may or may not be triggered by attacks. We consider four different models: a 3-layer GRUs [4], [9] model (marked as "model 1"), a 3-layer LSTM [10] network (marked as "model 2"), an encoder–decoder GRU network (marked as "model 3"), and an encoder–decoder LSTM network (marked as "model 4").

[12], [13], [14] are primarily designed and implemented for local use, thereby failing to incorporate the much-needed collaborative capacities inherent in IoT systems.

3) With the rapid evolution of IoT technology, known anomaly scenarios may only encapsulate a small fraction of potential cases. As a result, training a root cause analysis model to cover all possible anomaly scenarios proves challenging. Thus, the capacity for lifelong learning is essential for IoT root cause analysis systems to adapt to newly emerging anomaly scenarios.

Therefore, to develop a root cause analysis solution specifically tailored to meet the unique demands of IoT systems, we must answer three crucial questions: 1) How can we bolster the system's capacity for generalization? 2) How do we ensure efficient collaboration among local users, while simultaneously safeguarding their privacy? 3) How can we guarantee that the system has the ability to continually learn new knowledge and autonomously update itself to adapt to emerging anomaly scenarios?

B. Challenges and Contributions

The challenges posed by the aforementioned questions are multifaceted. First, the unpredictable nature of possible data fluctuations and the emergence of new data forms make it difficult for the model to discern the core pattern from the existing training data for each scenario. Classifiers typically rely on specific rules learned from the feature–label pairs of the training data to infer the most probable category for the test samples. However, such an approach could lead the model to focus on minor details excessively and even learn the noise inherent in individual instances [15], ultimately impeding the improvement of the model's generalization ability. Second, one possible method to preserve privacy is introducing noise into the data samples [16], [17], allowing local users to share their data for global model training without compromising privacy. However, the random nature of noise can disrupt useful patterns within the raw data and diminish test performance. Third, the process of acquiring new knowledge and self-renewal often renders the model temporarily inoperative [18], leading to significant inconvenience and potential safety risks in everyday life.

To address the above challenges, we propose a new framework named score-variational autoencoder (VAE) to analyze

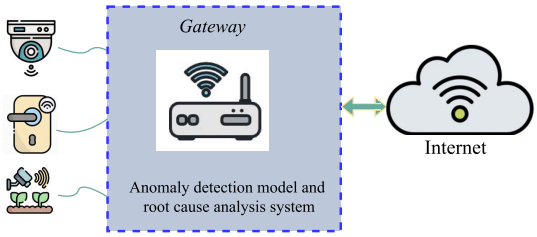


Fig. 3. Score-VAE at the gateway for root cause analysis.

the root cause of anomalies in IoT systems, which is deployed at the same place as the anomaly detection model, e.g., the gateway as in Fig. 3. Score-VAE incorporates a new training scheme and testing scheme that can fit the requirements of generalization, lifelong learning, collaboration, and privacy protection in root cause analysis of IoT systems. Specifically, we summarize the major contributions of this work as follows.

1) To bolster generalization, our system employs multiple global models collaborating in classification tasks. Each scenario has a dedicated global model trained for it, and the root causes of test samples are determined based on their reconstruction under each respective global model, rather than relying on a singular model deriving rules from feature–label pairs. This multimodal framework also enables the inclusion of newly emerging scenarios without disrupting existing ones.
2) We propose a novel training procedure to train a robust global model collaboratively without compromising privacy. In this method, the input provided to the local VAE network only includes distance relations of the scaled symbols derived from the original samples. Using the decoders of all local models and introducing noise, the server generates new samples to train the global models specific to each scenario.
3) Both the local and global models are realized using a dynamic VAE network, which adjusts the weights of different losses during training through techniques, such as stop gradient [19], [20] and multitask learning [21]. Furthermore, we introduce a Hamming distance-based test scheme, which helps alleviate noise's adverse effects.

TABLE I
CATEGORIZATION OF RELATED WORKS ON IDENTIFYING ROOT CAUSES WITH ML

Reference	*ML Technique*	*System*	*Fault Type*	*Root Cause Analysis Technique*
Chockalingam et al. [25]	Bayesian networks	Cyber-physical systems	Communication fault	Directed acyclic graphs
Wang et al. [26]	Bayesian networks	Thermal power plants	Alarm anomaly	Updating prior and posterior probabilities
Chen et al. [12]	Decision trees	Various systems	Software fault	Training with request traces and run-time information
Yang et al. [27]	Decision trees	Sensor networks	Sensor anomaly	Online random forest
Detzner et al. [28]	Decision trees	Manufacturing companies	Product fault	Interactive decision tree
Roelofs et al. [11]	Neural networks	Wind turbines	Wind turbine anomaly	Optimization algorithm applied to autoencoder network
Velásquez et al. [13]	Neural networks	Power transformers	Insulation anomaly	RL-based neural network classifier

4) We extensively evaluate the Score-VAE's performance using real-world IoT data. The experimental results demonstrate that Score-VAE effectively meets the demands of collaboration and privacy protection, thereby fitting well into the FL architecture. Compared to baseline models, Score-VAE exhibits superior generalization capacity and thus achieves improved test accuracy.

The remainder of this article is organized as follows. We introduce related work in Section II. Section III presents problem formulation and the overall architecture of Score-VAE. The training and test procedures of Score-VAE are introduced in Sections IV and V, respectively. We discuss the advantage of Score-VAE in Section VI and evaluate its performance in Section VII. This article is concluded in Section VIII.

II. RELATED WORK

Existing work of root cause analysis can be roughly divided into two categories: 1) identifying root causes with expertise and 2) identifying root causes with machine learning (ML).

A. Identify Root Causes With Expertise

In this category, expert knowledge is used to formulate specific rules [22], [23], [24] for determining the root cause of failures. Mourtzis et al. [22] designed a knowledge-based social networking app for employees to collaboratively diagnose problems during the development of large projects. Similarly, Mamoutova et al. [24] utilized expert knowledge to analyze failures of storage systems.

Techniques in this category demonstrate outstanding performance by leveraging specialized knowledge in fault classification or identification. However, we lack the expertise to establish effective rules in our particular case. Unlike traditional Internet applications, the IoT is an emerging industry that has experienced rapid growth in recent years. Its relatively short development history, fast product turnover, and heterogeneous nature pose significant challenges in acquiring the expertise needed to identify various IoT anomaly scenarios.

B. Identify Root Causes With ML

The identification of root causes typically relies on three commonly used ML techniques: 1) Bayesian networks; 2) decision trees; and 3) neural networks. To provide a better understanding of the contributions made by the related works, Table I categorizes them based on their main distinctive axes.

1) Bayesian Networks: Bayesian networks have been widely applied to represent the relationship between alarms and causes. Chockalingam and Katta [25] proposed a new framework for analyzing the root causes of problems observed in cyber-physical systems. However, conditional probability tables in directed acyclic graphs are difficult to obtain efficiently. Wang et al. [26] designed an approach to analyzing the underlying causes of a special type of alarm in thermal power plants. They updated the prior and posterior probabilities in the Bayesian network with data samples and determined the root cause from the set of parent nodes with the largest posterior conditional probability. However, it requires prior knowledge to be reliable.

2) Decision Trees: Chen et al. [12] introduced a decision tree-based approach for failure diagnosing. They trained the decision tree by using the request traces and run-time information from time periods when user-visible failures were presented. Yang et al. [27] devised an online random forest to diagnose sensor failures. Detzner et al. [28] employed an interactive decision tree for root-cause analysis of product detection in manufacturing companies. While decision trees are good at interpreting recognized causes, they are inefficient on time series data, and errors may increase fast when the number of categories is large.

3) Neural Networks: Roelofs et al. [11] introduced a method called anomaly root cause analysis. Their method could find those features contributing to wind turbine anomalies by applying an optimization algorithm to an autoencoder network. However, this approach assumes that the anomaly detection system has no FPs, which may not be possible in reality. Velásquez and Lara [13] adopted a reinforcement learning-based neural network classifier to identify the faults in power transformers. The existing works [11], [13], [14], [29], [30] primarily focus on identifying root causes of confirmed failures or attacks, overlooking the essential requirements of generalization, lifelong learning, collaboration, and privacy protection needed for root cause analysis in IoT systems.

This work falls under the second category and is implemented using the neural network technique. Different from existing work, our method involves anomalous scenarios, not only malicious attacks but also FPs caused by environmental changes. Moreover, unlike previous methods that locally train and apply classifiers, our approach incorporates a privacy-preserving training scheme and a hamming test scheme, enabling users to collaboratively train a global root cause analysis model with improved generalization and lifelong learning capabilities.

III. Problem Formulation and Overview of Score-VAE

A. Problem Formulation

Let $\mathbf{Q} = \mathbf{Q}_1, \mathbf{Q}_2, \ldots, \mathbf{Q}_N$ be a set of N anomaly samples, where each $\mathbf{Q}_i$ is a feature vector representing an anomaly in the IoT system. Let $\mathbf{Y} = \mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N$ be the corresponding set of root cause labels, where each $\mathbf{y}_i$ represents the label of the root cause of the anomaly sample $\mathbf{Q}_i$. Root cause analysis aims to learn a function $f : \mathbf{Q} \rightarrow \mathbf{Y}$ that maps each anomaly sample to its corresponding root cause label.

To achieve this, we can formulate root cause analysis as a supervised learning problem. Given a training data set $\mathcal{D} = (\mathbf{Q}_1, \mathbf{y}_1), (\mathbf{Q}_2, \mathbf{y}_2), \ldots, (\mathbf{Q}_N, \mathbf{y}_N)$, where each pair $(\mathbf{Q}_i, \mathbf{y}_i)$ represents an anomaly sample and its root cause label, we formulate an optimization problem. The objective is to minimize a loss function that quantifies the discrepancy between the predicted root cause labels $f_\theta(\mathbf{Q}_i)$ and the actual root cause labels $\mathbf{y}_i$ for all training samples. The optimization problem can be stated as follows:

$$\min_{\theta} \sum_{i=1}^{N} \mathcal{L}(f_\theta(\mathbf{Q}_i), \mathbf{y}_i) \tag{1}$$

where $\mathcal{L}$ is the loss function that measures the discrepancy between the predicted and actual root cause labels. The goal is to find the optimal parameters θ^* that minimize the overall loss across all training samples. The choice of the loss function depends on the specific problem and the desired properties of the root cause analysis model.

The solution to this optimization problem provides the learned model f_{θ^*}, which can be used to predict the root cause labels for anomalies. While the abstract form of the optimization problem is easy to follow, it is challenging to learn the mapping function f and its optimal parameters.

B. General Idea and Design Rationale

We propose Score-VAE to solve the above problem. For each anomaly scenario c ($1 \leq c \leq C$, C is the number of anomaly scenarios), Score-VAE trains a global VAE root cause analysis model $f_{\theta_c^*}$. Doing this, the original optimization problem can be stated as C subproblems. The detail of the loss function is defined in Section IV-B2. A privacy-preserving training procedure is introduced in Section IV to obtain θ_c^* ($1 \leq c \leq C$).

For a testing sample identified as an anomaly, every global VAE root cause analysis model θ_c^* ($1 \leq c \leq C$) will produce a reconstructed sample. We determine the root causes of these anomaly samples by calculating the Hamming distance between the reconstructed sample and the original test sample. The details of our test scheme are provided in Section V.

We design Score-VAE as an add-on service to complement the functionality of the existing FL-based anomaly detection system based on the following reasons.

1) It is common for anomaly detection models to rely mainly on benign data for training purposes [4]. This is due to the imbalance between benign and attack data, or the lack of access to (unknown) attack data. Such an approach is built on the widely accepted assumption that device manufacturers act in good faith and that IoT devices can be trusted upon initial release. While this strategy facilitates model training, it can introduce a higher risk of FPs. Consequently, we may need to frequently retrain the anomaly detection model for it to work well.
2) Retraining an existing anomaly detection model requires substantial overhead in both time and cost. Even though machine unlearning methods [31], [32], [33] may be used to speed up the model update, it is still very time consuming and undesirable to update the global anomaly detection model for better accuracy whenever a new scene (e.g., network congestion) is observed in local gateways.
3) Root cause analysis and global anomaly detection can use different modeling techniques, and the model for root cause analysis can be trained independently from that for anomaly detection. If we train Score-VAE incrementally based on new scenarios observed in local gateways, we can have great flexibility for extending the Score-VAE model. Whenever a new scene is observed, we can collect the corresponding data and use them to update the root cause analysis without touching the global anomaly detection. This benefit is further illustrated in the evaluation Section VII-B.

C. Architecture and Workflow

Fig. 4 provides a view of the Score-VAE architecture and its training process. Score-VAE is rooted in a FL architecture, which forms the basis of our innovative training process. This scheme is designed to generate new training samples for the model by introducing noise to the decoders of localized models submitted by client participants involved in each specific training scenario. In more detail, for every unique

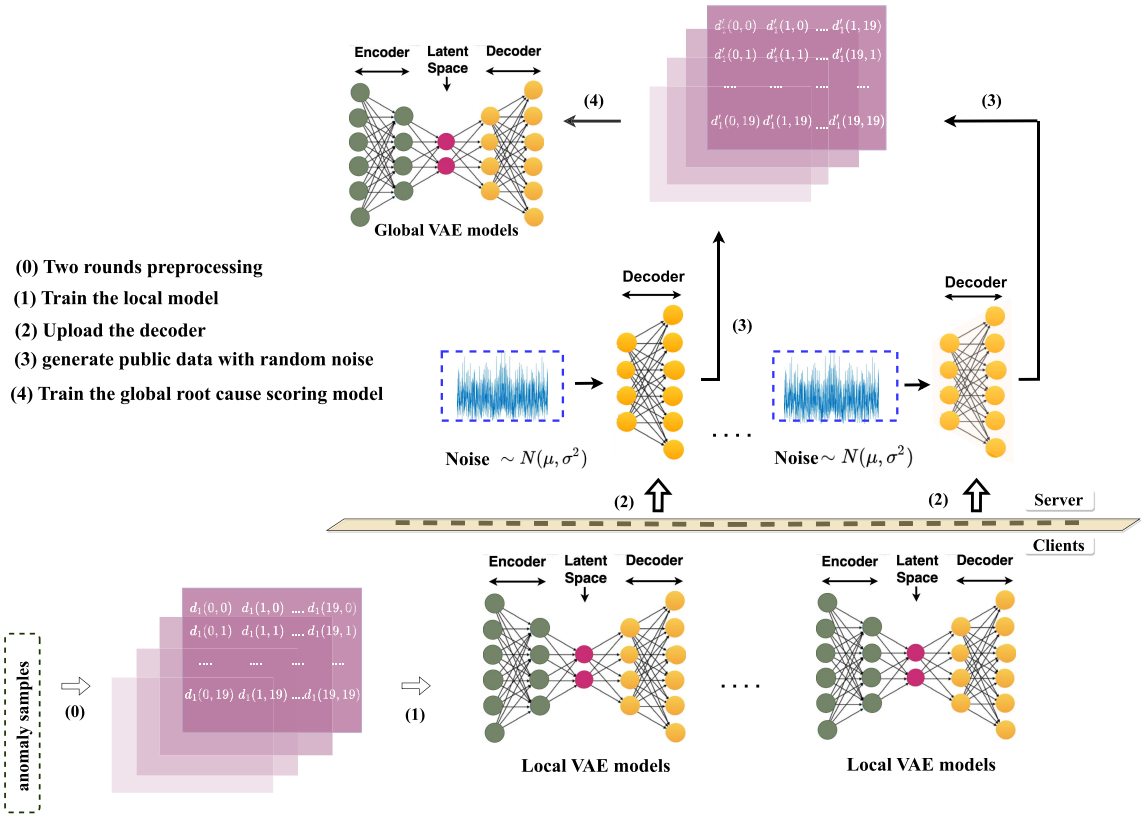


Fig. 4. Architecture of Score-VAE: the black arrows denote the training flow.

anomaly scenario, individual clients train a localized VAE generation model, subsequently uploading the decoder part of the model to the server. This server then harnesses these decoders to produce public data by injecting noise. The generated public data is then utilized to train a global root cause analysis model, which is explicitly tailored to the anomaly scenario at hand. Both local and global models are efficiently achieved through our dynamic VAE networks. The operational flow for the Score-VAE training is depicted in Fig. 4, with a more comprehensive discussion of the procedures available in Section IV.

During the testing phase, Score-VAE interacts with the existing anomaly detection system as follows: the test samples, or incoming traffic, are initially processed by the anomaly detection model. This model triggers alerts whenever anomalies are identified. These flagged anomaly testing samples are subsequently directed through all the globally trained VAE root cause analysis models for reconstruction. The global model whose reconstruction sample demonstrates the smallest Hamming distance with the original test sample is selected. The anomaly scenario associated with this selected global model is considered the root cause for the given test sample. The above procedure is further elaborated in Section V.

IV. Training Procedure of Score-VAE

For ease of reference, the main notations used for the training and testing of Score-VAE are listed in Table II. We introduce the training and the testing procedures of Score-VAE in this section and the next section, respectively.

We present a joint noise[2] privacy-preserve training scheme that can collaboratively train a powerful global VAE root cause analysis model while preserving clients' privacy. As shown in Fig. 4, our training scheme consists of five steps.

1) *Step 0:* Perform two rounds of preprocessing on the training data.
2) *Step 1:* Each client trains the local VAE generative model using the dynamic VAE network and the client's private data.
3) *Step 2:* All clients upload their decoder of the local VAE generative model trained in the first step to the server.
4) *Step 3:* the server uses random noise $g \sim \mathcal{N}(0, 1)$ to generate new data points from all the received local VAE generative models. The newly generated data can be treated as the server's public data.
5) *Step 4:* The server trains its global VAE root cause scoring model for each anomaly scenario using public data belonging to that scenario. The training method of the global VAE root cause scoring model is the same as the training method in step 2.

The details of the two rounds of data preprocessing are given in the next section. After that, we explain the details of dynamic VAE networks used to train the local and global root cause analysis models.

[2]We use the term "joint noise" because, with the help of noise, clients can jointly train a powerful model while ensuring privacy.

TABLE II
MAIN NOTATIONS

Notation	*Definition*
C	The number of anomaly scenarios
B	The number of batches
E	The number of epochs
D_j	The jth original anomaly sample
$\hat{D}_j$	The scaled value of the anomaly sample D_j
X_j	The converted matrix of the scaled sample $\hat{D}_j$
$\hat{f}_j^m$	The mth feature of the scaled data sample $\hat{D}_j$
$d_j(m,n)$	The absolute distance between $\hat{f}_j^m$ and $\hat{f}_j^n$ in the scaled data sample $\hat{D}_j$
$X_j^{'}$	The reconstruction matrix of the matrix sample X_j
θ_c^*	The global model's parameters of scenario c
$p(z_j\|X_j)$	The posterior distribution of the latent variable z_j
$q(X_j^{'})$	The distribution of reconstruction matrix $X_j^{'}$
$p(X_j)$	The distribution of original matrix X_j
$\mathcal{L}_j^{rec}$	The reconstruction loss of the anomaly sample D_j
$\mathcal{L}_j^{kl}$	The KL loss of the anomaly sample D_j
$\mathcal{L}_j^{dis}$	The distribution loss of the anomaly sample D_j
γ	The root cause threshold
s^c	The score that the anomaly is caused by the root cause c

* This table includes main notations used for the training process and the testing process.

TABLE III
PACKET CHARACTERISTICS USED IN SYMBOL MAPPING [4]

ID	*Characteristic*	*Value*
c_1	direction	1 = incoming, 0 = outgoing
c_2	local port type	bin index of port type
c_3	remote port type	bin index of port type
c_4	packet length	bin index of packet length
c_5	TCP flags	TCP flag values
c_6	protocols	encapsulated protocol types
c_7	IAT bin	bin index of packet inter-arrival time (IAT)

A. Two-Round Data Processing

1) First Round: We built an IoT testbed to emulate various IoT scenarios and collect data. We then map each packet to a symbol based on the seven characteristics of the packet listed in Table III. Specifically, the mapping assigns each unique combination of packet characteristics $(c_1, \ldots, c_7)$ a dedicated symbol representing the "type" of the particular packet [4]. Note that the symbols are indexed, and it is the index numbers that are used for model training. Thus, the sequence of packets can be translated into a sequence of symbols. We then adopt a moving window that covers 20 packets[3] For each packet, we use its symbol as the labeled class and the symbol sequence in its proceeding 20 packets as its feature. In this way, we can translate each packet into a sample.

2) Second Round: We conduct the second round of preprocessing on the data samples to facilitate training and protect privacy. First, we use min-max scaling [34] to scale each sample's feature values to a small range (A, B), where A and B are two hyperparameters, whose settings can be found in the later evaluation section. Specifically, denote a sample $D_j = (f_j^0, f_j^1, \ldots, f_j^{19})$, where $f_j^i (i = 0, \ldots, 19)$ are the 20 feature values. Denote $f_{\max}^i = \max\{f_0^i, f_1^i, \ldots\}$ over all samples, and $f_{\min}^i = \min\{f_0^i, f_1^i, \ldots\}$ over all samples. Then each sample $D_j = (f_j^0, f_j^1, \ldots, f_j^{19})$ is scaled to $\hat{D}_j = (\hat{f}_j^0, \hat{f}_j^1, \ldots, \hat{f}_j^{19})$, where $\hat{\mathbf{f}}_{c,j}^i = [(f_j^i - f_{\min}^i)/(f_{\max}^i - f_{\min}^i)](B - A) + A$. After that, we convert the input features $(\hat{f}_j^0, \hat{f}_j^1, \ldots, \hat{f}_j^{19})$ of the transformed data sample $\hat{D}_j$ into a matrix X_j, denoted as

$$X_j = \begin{Bmatrix} d_j(0,0) & d_j(1,0) & \cdots & d_j(19,0) \\ d_j(0,1) & d_j(1,1) & \cdots & d_j(19,1) \\ \cdots & \cdots & \cdots & \cdots \\ d_j(0,19) & d_j(1,19) & \cdots & d_j(19,19) \end{Bmatrix}$$

where $d_j(m,n)$ $(0 \le m, n \le 19)$ represents the absolute distance between $\hat{f}_j^m$ and $\hat{f}_j^n$ in the transformed data sample $\hat{D}_j$. Then the matrices will be used to train Score-VAE. The above processing steps make the actual inputs of the local VAE network only contain the distance relations of the scaled symbols in the original samples. By further introducing a joint noise training scheme, we can protect users' privacy.

To better illustrate our testing process (as shown Fig. 6), we, in the rest of this article, treat matrix X_j as an image of size 20×20 pixels, where $d_j(m,n)$ $(0 \le m, n \le 19)$ denote the pixel values. In other words, each sample is represented by a small image.

B. Dynamic VAE Network

As mentioned, the local and global root cause analysis models are realized through the dynamic VAE network. Dynamic VAE network means that the weights for different losses change dynamically during training.

1) Architecture: The architecture of the dynamic VAE network is shown in Fig. 5. VAE [35], [36], [37] network consists of three parts: 1) encoding; 2) reparameterization; and 3) decoding. Given a matrix sample X_j, the encoder approximates the latent variable's posterior distribution $p(z_j|X_j)$. The posterior distribution is assumed to be a normal distribution $\mathcal{N}(\mu_j, \sigma_j^2)$ with mean μ_j and standard deviation σ_j. Then we use reparameterization, which samples latent variable from the posterior distribution $p(z_j|X_j)$ and feeds it into the decoder to obtain the reconstructed sample X_j'.

2) Loss Function: It is worth noting that we use a dynamic loss function in the VAE network to replace the traditional loss function. The detail of the training process of our dynamic VAE network is illustrated in Algorithm 1. Specifically, for

[3]The window size of 20 packets for feature representation is empirical and was validated to be effective for anomaly detection in [4].

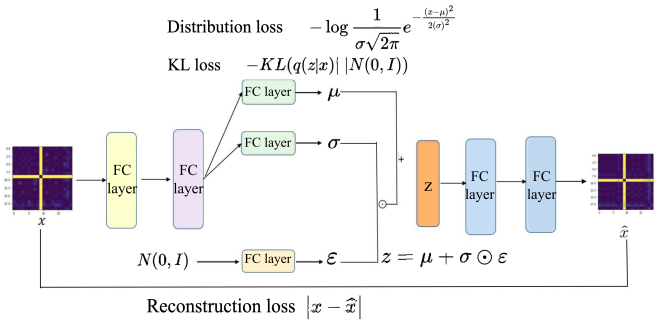


Fig. 5. Architecture of dynamic VAE network in Score-VAE.

each training sample j in batch b, we calculate its reconstruction loss $\mathcal{L}_j^{rec}$, kl loss $\mathcal{L}_j^{kl}$, and distribution loss $\mathcal{L}_j^{\text{dis}}$. The reconstruction loss $\mathcal{L}_j^{rec}$ is the L_1 norm distance $|X_j - X_j'|$ of the reconstruction sample X_j' and the original sample X_j. The kl loss $\mathcal{L}_j^{kl}$ represents the KL divergence $(1/2)(2\log(\sigma_j) + 1 - \sigma_j^2 - \mu_j^2)$ between the posterior distribution $p(z_j|X_j)$ of the latent variable and the standard normal distribution $\mathcal{N}(0, 1)$. The distribution loss is determined by the probability that the original sample X_j belongs to the posterior distribution $p(z_j|X_j)$ of the latent variable, denoted as

$$\mathcal{L}_j^{\text{dis}} = -\log\left(p\left(X_j|\mu_j, \sigma_j^2\right)\right) = -\log\left(\frac{1}{\sqrt{2\pi}\sigma_j} e^{-\frac{(X_j-\mu_j)^2}{2\sigma_j^2}}\right).$$

3) Algorithm Details: We calculate the total reconstruction loss, kl loss, and distribution loss for each batch b of J samples, i.e., $\mathcal{L}^{rec}(b) = \sum_{j=0}^{J-1} \mathcal{L}_j^{rec}$, $\mathcal{L}^{kl}(b) = \sum_{j=0}^{J-1} \mathcal{L}_j^{kl}$ and $\mathcal{L}^{\text{dis}}(b) = \sum_{j=0}^{J-1} \mathcal{L}_j^{\text{dis}}$. Similar to [38], we adopt dynamic loss $\mathcal{L}^{\text{SUM}}(b)$, which is the weighted sum of the reconstruction loss, the kl loss, and the distribution loss, denoted as $\mathcal{L}^{\text{SUM}}(b) = w_{rec,kl}(\mathcal{L}^{kl}(b) + \mathcal{L}^{rec}(b)) + C_1 w_{\text{dis}} \mathcal{L}^{\text{dis}}(b) + C_2 \log(v_{rec,kl} v_{\text{dis}})$, where $v_{\text{dis}} = 2w_{\text{dis}}{}^{-1/2}$. We add a constraint term $C_2 \log(v_{rec,kl} v_{\text{dis}})$ to avoid the weights becoming too small and use the hyperparameters C_1 and C_2 to adjust the order of magnitude of different terms. The weights of different losses can be automatically tuned with model parameters through gradient descent, as shown in Algorithm 1. The reconstruction loss and the kl loss share the same weight $w_{rec,kl}$ because they are losses in the traditional VAE network. Then we check whether the weight of the loss is out of the defined range (lines 9 – 11 in Algorithm 1). However, this process may make the loss calculation process discrete, and we cannot calculate the gradient. To solve the problem, we use an intermediate vector and the stop gradient method as shown from lines 12 to 14 in Algorithm 1. Specifically, we use the PyTorch *.detach()* method to stop gradient [19], [20]. During forward propagation, the *.detach()* method does not make any changes. We compute the loss of $v_{\text{dis}} = a_{\text{dis}} + (v_{\text{dis}} - a_{\text{dis}})$. During back propagation, the part with *.detach()* does not compute the gradient, so we only compute the gradient of $v_{\text{dis}} = a_{\text{dis}}$. Note that a_{dis} is continuous, and its gradient can be calculated.

4) Explanation of Distribution Loss: The purpose of introducing the distribution loss is to maximize the probability that X_j belongs to the posterior distribution $q(X_j'|z_j)$ of the reconstruction sample X_j'.

The goal of reconstruction loss is to minimize the distance $|X_j - X_j'|$, where X_j' is the reconstruction sample and X_j is the original sample. The goal of the Kullback–Leibler loss is to make sure the distribution of the latent variable z_j follows the standard normal distribution. We denote the distribution loss as $L_j^{\text{dis}} = -\log(P(X_j|z_j)$. So minimizing the distribution loss is equal to minimizing the distance of $q(X_j')$ and $p(X_j)$, where $q(X_j')$ is the distribution of reconstruction data point X_j' and $p(X_j)$ is the distribution of original data point X_j.

Algorithm 1: Dynamic VAE Network

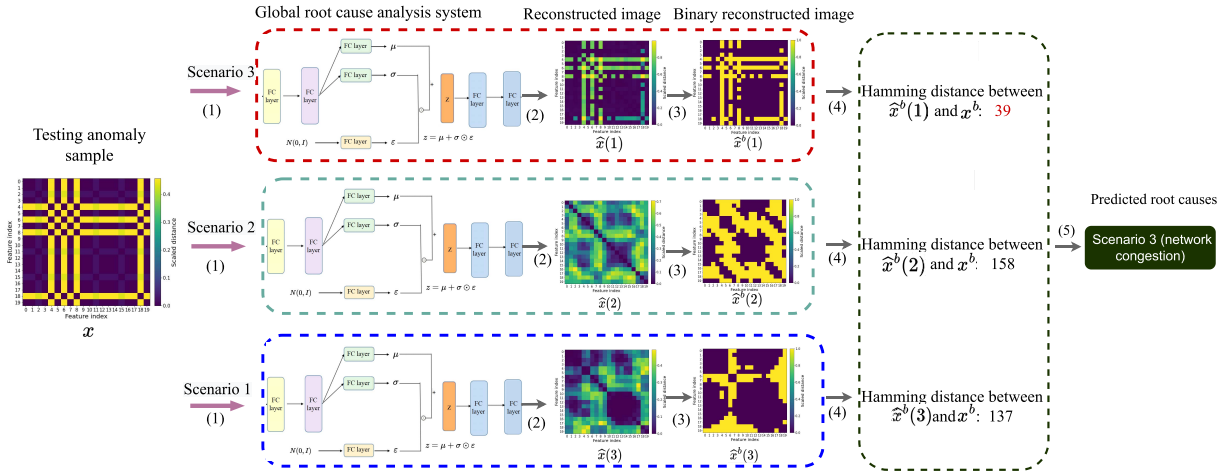
Input: Training set $\mathbf{X}$, $(a_{rec,kl}, a_{dis})$, $[h_{rec,kl}^{\min}, h_{rec,kl}^{\max}]$, $[h_{dis}^{\min}, h_{dis}^{\max}]$, mini-batch size J
Output: Optimal parameters θ^* for the VAE model
1 Randomly initialize the parameters of the VAE model
2 **repeat**
3 **for** *Batch* $b \leftarrow 1$ **to** B **do**
4 $\mathcal{L}^{rec}(b) = \sum_{j=0}^{J-1} |X_j(b) - X_j'(b)|$
5 $\mathcal{L}^{kl}(b) = \sum_{j=0}^{J-1} \frac{1}{2}(2\log(\sigma_j(b)) + 1 - \sigma_j^2(b) - \mu_j^2(b))$
6 $\mathcal{L}^{dis}(b) = \sum_{j=0}^{J-1} -\log(\frac{1}{\sigma_j(b)\sqrt{2\pi}} e^{-\frac{(X_j(b)-\mu_j(b))^2}{2\sigma_j^2(b)}})$
7 $v_{rec,kl} = a_{rec,kl}$
8 $v_{dis} = a_{dis}$
9 // Check loss weight range
10 $v_{rec,kl} = \text{Compare}(a_{rec,kl}, h_{rec,kl}^{\min}, h_{rec,kl}^{\max})$
11 $v_{dis} = \text{Compare}(a_{dis}, h_{dis}^{\min}, h_{dis}^{\max})$
12 // Use.detach() to stop gradient during backward
13 $v_{rec,kl} = a_{rec,kl} + (v_{rec,kl} - a_{rec,kl}).detach()$
14 $v_{dis} = a_{dis} + (v_{dis} - a_{dis}).detach()$
15 $w_{rec,kl} = \frac{1}{2v_{rec,kl}^2}$
16 $w_{dis} = \frac{1}{2v_{dis}^2}$
17 $\mathcal{L}^{SUM}(b) = w_{rec,kl}(\mathcal{L}^{rec}(b) + \mathcal{L}^{kl}(b)) + C_1 w_{dis} \mathcal{L}^{dis}(b) + C_2 \log(v_{rec,kl} v_{dis})$
18 Backward $\mathcal{L}^{SUM}(b)$ to update the model parameters and loss weights
19 **until** *Convergence*;
20 **return** *Optimal model parameters* θ^*;
21 **Function** `Compare(`*a*, min, max`)`**:**
22 result;
23 **if** $a \leq$ min **then**
24 result = min;
25 **else if** $a \geq$ max **then**
26 result = max;
27 **else**
28 result = a;
29 **return** result;

C. Time Complexity of the Training Process

The time complexity of training Score-VAE is as follows:

1) *Encoding Phase:* $O(E(M/B)((K-1)N^2 + NL + IN))$.
2) *Reparameterization Phase:* $O(E(M/B)L)$.
3) *Decoding Phase:* $O(E(M/B)((K-1)N^2 + NI + NL))$.
4) *Loss Computation:* $O(E(M/B)BL) = O(EML)$.
5) *Overall Time Complexity:* $O(E(M/B)((K-1)N^2 + NI + NL) + EML)$

where E is the number of epochs, M is the number of samples in the data set, B is the batch size, K is the number of layers in the encoder, I is the input and output size, L is the size of



(1) Pass the test image samples to the root cause analysis system. (2) Reconstruct test image samples with the global root cause analysis model for each scenario and output reconstructed images.

(3) Convert each reconstructed image into a binary image. (4) Calculate the Hamming distance between the reconstructed binary image of each scenario and the binary image of original test sample.

Fig. 6. Hamming distance between the binary reconstructed image and the original binary image in the network congestion test example.

the latent space, and N is the number of units in each hidden layer.

We assume that the time complexity of each hidden layer in the encoder and decoder is of the same order of magnitude and the number of layers in the encoder and decoder part is the same. However, it is worth noting that while these assumptions may generally hold true, they may not be applicable in all cases. The first, second, and third terms represent the time complexity of processing the input through the encoder network, reparameterization, and processing the latent space through the decoder network, respectively. The fourth term represents the time complexity of computing the loss function during training, which depends on the number of iterations performed during training. The overall time complexity of Score-VAE considers all four phases: encoding, reparameterization, decoding, and loss computation. Note that encoding and decoding have the same time complexity in this context.

V. Test Procedure of Score-VAE

Hamming test: During the testing period, Score-VAE works together with the existing anomaly detection model. The test samples first pass through the anomaly detection model, which raises alarms whenever anomalies are detected. Then the anomalies are passed to the global root cause analysis model for root cause analysis.

Fig. 6 uses an example to illustrate the flow of our test scheme, which includes five steps.

1) *Step 0:* Convert the anomaly test data into image samples using the same preprocessing method described in Section IV-A.
2) *Step 1 [Refer to (1) in Fig. 6]:* Pass the test image samples to the root cause analysis system.
3) *Step 2 [Refer to (2) in Fig. 6]:* Reconstruct test image samples with the global root cause analysis model for each scenario and output reconstructed images.
4) *Step 3 [Refer to (3) in Fig. 6]:* Convert each reconstructed image into a binary image. The basic idea is to calculate the average of all pixel values in an image. Set a pixel to 1 if it is greater than the mean, and 0 otherwise.
5) *Step 4 [Refer to (4) in Fig. 6]:* Calculate the Hamming distance between the reconstructed binary image of each scenario and the binary image of the original test sample. The Hamming distance [39] represents the number of pixels of two images at which their corresponding values are different.
6) *Step 5 [Refer to (5) in Fig. 6]:* Compare the Hamming distance between the reconstructed binary image and the original binary image. The scenario that reconstructs the binary image with the smallest Hamming distance will be treated as the root cause of the anomaly sample.

In each group, if the proportion of samples caused by root cause c is greater than the root cause threshold γ, the root cause c is considered to be one of the root causes for that group. Finally, the global VAE root cause scoring system outputs its root cause score for the group

$$s = \begin{bmatrix} s^1 & s^2 & \dots & s^C \end{bmatrix}^T \tag{2}$$

where s^c $(1 \le c \le C)$ denotes the score that the anomaly is caused by the root cause c. The score s^c is set to 1 if c is one of the anomaly root causes of the group, otherwise set to 0.

VI. Further Discussion (What Are the Main Advantages of Score-VAE for Root Cause Analysis?)

A. Better Generalization Capability

Score-VAE exhibits superior generalization capabilities due to its reliance on the reconstruction effect of various scenario-specific global VAE root cause scoring models. This

approach enables the model to assess the category of a test sample by considering its overall reconstruction, rather than relying solely on a single model to directly learn rules from the mapping relationship between feature–label pairs.

B. Lifelong Learning

Moreover, this training scheme ensures the independence of global models for different scenarios within the root cause analysis system. As a result, training a new global model for a newly observed scene does not temporarily render the existing global models unavailable.

C. Privacy Preserving

The privacy of local users can be effectively preserved for the following reasons.

1) The processing steps in the system ensure that the actual inputs of the local VAE network only contain distance relations of scaled symbols found in the original samples.
2) Only the decoder parts of the network are uploaded to the server, ensuring that sensitive information remains on the local device.
3) The data samples used to train the global model are generated from random noise, further safeguarding user privacy.

We mitigate the potential negative impacts arising from the randomness of noise and the uncertainty of noise addition methods with the following measures.

1) The noise employed in the system exhibits a distribution similar to that of the latent variable, enhancing its compatibility with the training process.
2) The generative ability of the VAE network enables the establishment of a connection between random noise and the training data, facilitating meaningful learning.
3) The dynamic VAE network allows the trained model not only to reconstruct the pixel values of the sample but also to capture the underlying distribution of the data.
4) By converting the reconstructed image to a binary image using the Hamming distance, the system effectively reduces the distance between matching pixels while enlarging the distance between unmatched pixels. This process enables the identification and counting of flipped data bits only in the binary image, thereby minimizing potential privacy breaches.

VII. Experimental Evaluation

We perform a series of experiments to illustrate the benefits of Score-VAE by answering the following research questions (RQs). *RQ1*: Can Score-VAE correctly identify the causes that trigger the alarms from the anomaly detection model? *RQ2*: What is the performance of Score-VAE compared to the baselines? *RQ3*: How does each component of Score-VAE contribute to the overall performance?

A. Data Preparation

The false alarms can be caused by attacks or the temporal-spatial dynamics of the network environment. However, we did not find any existing IoT system data sets containing all kinds of traffic simultaneously. Hence, we built an IoT testbed with smart cameras (model DLink DCS-932LB) to emulate various IoT traffic scenarios. The cameras are connected to the Internet via a gateway (as shown in Fig. 1), and we use sniffer software to capture the data package going through the gateway via port mirroring.

We collected both malicious data and benign data. The malicious data include data flows under different Mirai attack stages [40]: loading, scanning, and Distributed Denial of Service (DDoS) attack. Benign data includes data flow under nonattack conditions, including different network configurations, e.g., different maximum transmission unit (MTU) values, and different network traffic loads, e.g., temporary network congestion. The details are as follows.

1) *Set_1:* Mirai data includes malicious traffic during the Mirai loading phase. Specifically, Mirai infects a smart camera, turning it into a remotely controlled "zombie." This set includes 64 101 samples.
2) *Set_2:* Mirai data includes malicious traffic during the Mirai scanning stage. A bot sends a small amount of data to declare its connection with the command-and-control (CnC)[4] server, and the CnC server sends the bots some commands. This set includes 18 494 samples.
3) *Set_3:* Mirai data includes malicious traffic during the DDoS attack stage. A bot attacks a selected target by generating a large amount of fake data. This set includes 20 310 samples.
4) *Set_4:* It includes benign traffic when network configuration changes. We changed the TCP maximum segment size (MSS) to different values to test this scenario. Note that TCP may dynamically change MSS to suit the path MTU (PMTU). This set includes 34 971 samples.
5) *Set_5:* Network congestion data includes traffic in the presence of network congestion. Specifically, we captured live video data from the cameras while other local machines were streaming Internet videos. In this case, we observed frequent packet retransmissions and losses, which manifested the network congestion. This set includes 20 102 samples.
6) *Set_6:* Normal data includes benign traffic when the cameras are in normal working mode and the local network traffic load from other machines is light. This set includes 20 728 samples.

Note that Set_6 is used only for training a 3-layer gated recurrent unit (GRU) anomaly detection model [4]. This anomaly detection model detects anomalies and reports alarms to Score-VAE, which analyzes the root causes. Set_i $(1 \leq i \leq 5)$ are for training/testing Score-VAE. We divide each Set_i $(1 \leq i \leq 5)$ into training set Train_i (80%), validation set Validation_i (10%), and testing set Test_i (10%).

[4]The CnC server is a computer controlled by an attacker to send commands to systems compromised by Mirai malware. Since we have control of the Mirai bots, we configure the IP address of the CnC server to our controlled machine to avoid attacking any machine beyond our lab.

TABLE IV
OVERALL PERFORMANCE OF SCORE-VAE SYSTEM

Test traffic	No.	Actual root cause	Probability distribution of identified root causes (%)				Test accuracy (%)
			Network congestion	Network configuration change	Mirai attack	Others	
Malicious	1	Mirai (loading)	1.49	0	98.51	0	98.51
	2	Mirai (scanning)	0	0	100	0	100
	3	Mirai (DDoS attack)	0	0	100	0	100
Benign	4	Network configuration change	0	99.05	0	0.95	99.05
	5	Network congestion	100	0	0	0	100

TABLE V
OVERALL PERFORMANCE OF DECISION TREE (BASELINE 1)

Test traffic	No.	Actual root cause	Probability distribution of identified root causes (%)				Test accuracy (%)
			Network congestion	Network configuration change	Mirai attack	Others	
Malicious	1	Mirai (loading)	0	100	0	0	0
	2	Mirai (scanning)	0	0	100	0	100
	3	Mirai (DDoS attack)	0	100	0	0	0
Benign	4	Network configuration change	0	100	0	0	100
	5	Network congestion	100	0	0	0	100

TABLE VI
OVERALL PERFORMANCE OF THE THREE-LAYER LSTM MODEL (BASELINE 2)

Test traffic	No.	Actual root cause	Probability distribution of identified root causes (%)				Test accuracy (%)
			Network congestion	Network configuration change	Mirai attack	Others	
Malicious	1	Mirai (loading)	64.66	0	35.34	0	35.34
	2	Mirai (scanning)	0	0	100	0	100
	3	Mirai (DDoS attack)	42.86	0	57.14	0	57.14
Benign	4	Network configuration change	0	100	0	0	100
	5	Network congestion	97.06	0	0	2.94	97.06

B. Answer to RQ1

We consider three anomaly scenarios: 1) network configuration changes (scenario 1); 2) Mirai attacks (scenario 2); and 3) network congestion (scenario 3). We use Train_4 to train the global model of scenario 1, Train_2 to train the global model of scenario 2, and Train_5 to train the global model of scenario 3. The data of $\text{Test}_1 \sim \text{Test}_5$ are used for testing.

We use the following hyperparameters. The batch size is 100 and learning rate is $0.5e^{-3}$. The values of C_1 and C_2 (refer to Section IV-B3) are set to 0.2 and 1000 respectively. The min-max scaling range value for the Mirai attack scenario is set to $(0, 1)$. The min-max scaling range value for the network congestion scenario is set to $(-2, 2)$. The min-max scaling range value for the network configuration changes scenario is set to $(-4, 4)$. The root cause threshold γ is set to 0.2.

Experimental Results: The overall performance of Score-VAE is shown in Table IV. The results show that Score-VAE can successfully identify the root causes of alarms with an accuracy higher than 98%.

C. Answer to RQ2

As mentioned in the related work section, other supervised learning models may be used for root cause analysis (for example, a simple neural network/tree-based model). To better illustrate the advantages of Score-VAE, we comprehensively compare Score-VAE with three baselines from the perspectives of test accuracy and privacy preservation.

1) *Baseline 1:* This baseline uses a decision tree classifier [41] for root cause analysis.
2) *Baseline 2:* This baseline uses a 3-layer long short-term memory (LSTM) neural network [10] to train a classifier for root cause analysis.
3) *Baseline 3:* This baseline uses an Autoencoder [42], [43] model for root cause analysis. This baseline also orchestrates multiple generative models to collaborate in classification tasks.

Experimental Results: In terms of test accuracy, Score-VAE can achieve the best performance, as shown in Tables IV–VII. This is due to Score-VAE's training and

TABLE VII
OVERALL PERFORMANCE OF THE AUTOENCODER MODEL (BASELINE 3)

Test traffic	No.	Actual root cause	Probability distribution of identified root causes (%)				Test accuracy (%)
			Network congestion	Network configuration change	Mirai attack	Others	
Malicious	1	Mirai (loading)	0	0	100	0	100
	2	Mirai (scanning)	0	0	100	0	100
	3	Mirai (DDoS attack)	0	0	100	0	100
Benign	4	Network configuration change	0	100	0	0	100
	5	Network congestion	0	0	100	0	0

TABLE VIII
COMPARISON OF SCORE-VAE WITH BASELINES

Method	Overall test accuracy	Privacy protection
Baseline 1	49.28%	No
Baseline 2	69.56%	No
Baseline 3	90.23%	No
Score-VAE	**99.14%**	**Yes**

test procedures enhancing the model's generalization ability.

The under performance of baseline 1 and baseline 2 can be attributed to the diverse and complex nature of data across various anomaly scenarios. Tables V and VI highlight a significant drawback of these baseline models—they struggle to accurately identify malicious test data that significantly deviate from the training set of malicious data. Essentially, this issue stems from a common problem in ML called overfitting. The models excessively adapt to the specific features and noise present in the training data, resulting in a decline in their predictive performance when faced with new, unseen data. We also evaluated alternative models, such as a three-layer GRU model, which exhibited similar limitations. The diversity and complexity of our data pose immense challenges in improving the generalization capacity of a single classifier model while maintaining effective feature learning. Furthermore, this restricts the model's ability to self-update when encountering additional anomaly scenarios in the future. Table VII reveals that while baseline 3 shows improvement compared to baselines 1 and 2, it still tends to classify congested traffic as Mirai attacks incorrectly.

Regarding privacy preservation, Score-VAE takes a strategic approach, which only transmits the decoder component of the local VAE generative model to the server. The server trains the global root cause scoring model by feeding noise to the decoder part to generate new samples. Thanks to the efficient fusion of FL architecture, preprocessing scheme, and joint noise training scheme, Score-VAE ensures the safeguarding of users' data privacy.

Table VIII summarizes the overall comparison results.

D. Answer to RQ3

Score-VAE includes three critical components: joint noise, dynamic VAE, and Hamming test. To demonstrate the contribution of each component, we perform an ablation study. The *baseline* is one that disables joint noise and uses the traditional VAE network and L_1 loss. Table IX shows the results.

1) Joint Noise: If we disable the joint noise, the system will lack adequate protection for privacy.

2) Dynamic VAE: The second row of Table IX indicates that dynamic VAE, working with the Hamming test, can significantly improve the accuracy of identifying network congestion. The main reason is that the dynamic loss function consists of traditional losses (reconstruction loss and *Kl* loss) and distribution loss. The distribution loss can help the VAE model learn the latent variables better and ensure that the posterior distribution of reconstructed data is close to the distribution of original data. In addition, the weights learned by the model can better capture the importance of different losses than average weights or manually adjusted weights.

3) Hamming Test: Comparing the fourth row and the first row of Table IX, we find that the Hamming test can improve the accuracy of identifying network configuration and network congestion. The main reason is that Hamming distance can reduce the distance of matching pixels and enlarge the distance of unmatched pixels by converting the reconstructed image to a binary image. Then only the incorrect data bits flipped in the binary image will be counted. Compared with other loss calculation methods, such as L_1 loss and *kl* divergence, Hamming distance reduces the noise error value brought by the preprocessing and reconstruction process.

In summary, the ablation study shows that combining these three components can significantly improve the accuracy of identifying nonmalicious root causes (e.g., network configuration change or network congestion), while achieving similar accuracy in identifying malicious root causes (only the accuracy for detecting Mirai loading is slightly reduced).

VIII. CONCLUSION AND FUTURE WORK

FL-based IoT anomaly detection systems often generate false-positive alarms due to the spatial-temporal dynamics within the FL architecture's network environment. For instance, temporary wireless link failures or network configuration changes can lead to duplicate TCP packet retransmissions. This article addresses this challenge by introducing a novel root cause analysis system called Score-VAE. Designed as an add-on service for existing FL-based IoT anomaly

TABLE IX
ABLATION STUDY OF COMPONENTS IN SCORE-VAE

Joint noise	Dynamic VAE	Hamming test	Test accuracy improvement*					Privacy protection*
			Mirai (loading)	Mirai (scanning)	Mirai (DDoS attack)	Network configuration change	Network congestion	
✓	✓	✓	**1.49%↓**	–	–	**99.05%↑**	**100%↑**	**Yes**
✗	✓	✓	10.10%↓	–	–	–	90.47%↑	No (same as the *baseline*)
✓	✗	✓	100%↓	58.07%↓	–	100%↑	3.44%↑	Yes
✓	✓	✗	–	–	–	–	–	Yes

* The improvement is over the *baseline* that disables joint noise and uses traditional VAE network and L_1 loss. "–" means the same as the *baseline*.

detection systems, Score-VAE effectively filters out FPs. It leverages the reconstruction and generation capabilities of the VAE network and integrates three crucial components: joint noise, dynamic VAE, and Hamming test. These components ensure strong generalization, lifelong learning, collaboration, and privacy protection capabilities for root cause analysis in IoT systems. Real-world IoT data evaluation demonstrates Score-VAE's ability to accurately identify the root causes of anomalies while preserving the privacy of local users.

However, Score-VAE shares a fundamental limitation with other VAE-based solutions: its knowledge of root causes is learned solely from known scenarios. Overcoming this limitation presents a challenging problem for future research. Our aim is to explore methods that enable Score-VAE to learn from unknown scenarios and further enhance its generalization abilities. Additionally, while inferred root causes can expedite the work of network operators, it is still crucial for them to verify these causes to ensure the network's safe operation. Consequently, another future work is improving the system to automatically verify root causes and preemptively filter out false alarms, which may be achieved by incorporating causal correlation and contextual monitoring. By accomplishing this, we can further reduce the work overhead for network operators.

REFERENCES

[1] G. Nebbione and M. C. Calzarossa, "Security of IoT application layer protocols: Challenges and findings," *Future Internet*, vol. 12, no. 3, p. 55, 2020.

[2] H. F. Atlam and G. B. Wills, "IoT security, privacy, safety and ethics," in *Digital Twin Technologies and Smart Cities*. Cham, Switzerland: Springer, 2020, pp. 123–149.

[3] Z. Ling, J. Luo, Y. Xu, C. Gao, K. Wu, and X. Fu, "Security vulnerabilities of Internet of Things: A case study of the smart plug system," *IEEE Internet Things J.*, vol. 4, no. 6, pp. 1899–1909, Dec. 2017.

[4] T. D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, and A.-R. Sadeghi, "DïoT: A federated self-learning anomaly detection system for IoT," in *Proc. IEEE 39th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, 2019, pp. 756–767.

[5] G. Bovenzi, G. Aceto, D. Ciuonzo, V. Persico, and A. Pescapé, "A hierarchical hybrid intrusion detection approach in IoT scenarios," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, 2020, pp. 1–7.

[6] J. Fan, "Machine learning and unlearning for IoT anomaly detection," Ph.D. dissertation, Dept. Comput. Sci., Univ. Victoria, Victoria, BC, Canada, 2023.

[7] B. B. Zarpelão, R. S. Miani, C. T. Kawakani, and S. C. de Alvarenga, "A survey of intrusion detection in Internet of Things," *J. Netw. Comput. Appl.*, vol. 84, pp. 25–37, Apr. 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804517300802

[8] S. Marchal, M. Miettinen, T. D. Nguyen, A.-R. Sadeghi, and N. Asokan, "AuDI: Towards autonomous IoT device-type identification," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1402–1412, Jun. 2019.

[9] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," 2014, *arXiv:1412.3555*.

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

[11] C. M. Roelofs, M.-A. Lutz, S. Faulstich, and S. Vogt, "Autoencoder-based anomaly root cause analysis for wind turbines," *Energy AI*, vol. 4, Jun. 2021, Art. no. 100065.

[12] M. Chen, A. X. Zheng, J. Lloyd, M. I. Jordan, and E. Brewer, "Failure diagnosis using decision trees," in *Proc. IEEE Int. Conf. Auton. Comput.*, 2004, pp. 36–43.

[13] R. M. A. Velásquez and J. V. M. Lara, "Root cause analysis improved with machine learning for failure analysis in power transformers," *Eng. Failure Anal.*, vol. 115, Sep. 2020, Art. no. 104684.

[14] A. Lokrantz, E. Gustavsson, and M. Jirstrand, "Root cause analysis of failures and quality deviations in manufacturing using machine learning," *Procedia CIRP*, vol. 72, pp. 1057–1062, Jun. 2018.

[15] T. Dietterich, "Overfitting and undercomputing in machine learning," *ACM Comput. Surveys*, vol. 27, no. 3, pp. 326–327, 1995.

[16] Q. Chen et al., "Differentially private data generative models," 2018, *arXiv:1812.02274*.

[17] M. Z. Islam and L. Brankovic, "Privacy preserving data mining: A noise addition framework using a novel clustering technique," *Knowl. Based Syst.*, vol. 24, no. 8, pp. 1214–1223, 2011.

[18] T. T. Nguyen, T. T. Huynh, P. L. Nguyen, A. W.-C. Liew, H. Yin, and Q. V. H. Nguyen, "A survey of machine unlearning," 2022, *arXiv:2209.02299*.

[19] M. Abadi et al., "{TensorFlow}: A system for {large-scale} machine learning," in *Proc. 12th USENIX Symp. Oper. Syst. Design Implement. (OSDI)*, 2016, pp. 265–283.

[20] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 8024–8035.

[21] R. Caruana, "Multitask learning," *Mach. Learn.*, vol. 28, no. 1, pp. 41–75, 1997.

[22] D. Mourtzis, M. Doukas, and N. Milas, "A knowledge-based social networking app for collaborative problem-solving in manufacturing," *Manuf. Lett.*, vol. 10, pp. 1–5, Oct. 2016.

[23] G. Fix, "An expert system for test failure root cause discovery," in *Proc. IEEE 7th Int. Conf. Inf. Technol. New Gener.*, 2010, pp. 1014–1019.

[24] O. V. Mamoutova, M. B. Uspenskiy, A. V. Sochnev, S. V. Smirnov, and M. V. Bolsunovskaya, "Knowledge based diagnostic approach for enterprise storage systems," in *Proc. IEEE 17th Int. Symp. Intell. Syst. Inf. (SISY)*, 2019, pp. 207–212.

[25] S. Chockalingam and V. Katta, "Developing a Bayesian network framework for root cause analysis of observable problems in cyber-physical systems," in *Proc. IEEE Conf. Inf. Commun. Technol.*, 2019, pp. 1–6.

[26] J. Wang et al., "Root-cause analysis of occurring alarms in thermal power plants based on Bayesian networks," *Int. J. Elect. Power Energy Syst.*, vol. 103, pp. 67–74, Dec. 2018.

[27] L. Yang, J. Zhang, F. Deng, and J. Chen, "Sensor fault diagnosis based on on-line random forests," in *Proc. IEEE 35th Chin. Control Conf. (CCC)*, 2016, pp. 4089–4093.

[28] A. Detzner, R. Rückschloß, and M. Eigner, "Root-cause analysis with interactive decision trees," in *Proc. IEEE 24th Int. Conf. Inf. Visual. (IV)*, 2020, pp. 322–327.
[29] N. G. Lo, J.-M. Flaus, and O. Adrot, "Review of machine learning approaches in fault diagnosis applied to IoT systems," in *Proc. IEEE Int. Conf. Control Autom. Diagn. (ICCAD)*, 2019, pp. 1–6.
[30] S. S. Tayarani-Bathaie, Z. S. Vanini, and K. Khorasani, "Dynamic neural network-based fault diagnosis of gas turbine engines," *Neurocomputing*, vol. 125, pp. 153–165, Feb. 2014.
[31] L. Bourtoule et al., "Machine unlearning," in *Proc. IEEE Symp. Security Privacy (SP)*, 2021, pp. 141–159.
[32] Y. Cao and J. Yang, "Towards making systems forget with machine unlearning," in *Proc. IEEE Symp. Security Privacy*, 2015, pp. 463–480.
[33] J. Brophy and D. Lowd, "Machine unlearning for random forests," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 1092–1104.
[34] F. Pedregosa et al., "Scikit-learn: Machine learning in python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Nov. 2011.
[35] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," 2013, *arXiv:1312.6114*.
[36] D. Park, Y. Hoshi, and C. C. Kemp, "A multimodal anomaly detector for robot-assisted feeding using an LSTM-based variational autoencoder," *IEEE Robot. Autom. Lett.*, vol. 3, no. 3, pp. 1544–1551, Jun. 2018.
[37] D. Park, Z. Erickson, T. Bhattacharjee, and C. C. Kemp, "Multimodal execution monitoring for anomaly detection during robot manipulation," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2016, pp. 407–414.
[38] A. Kendall, Y. Gal, and R. Cipolla, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 7482–7491.
[39] R. W. Hamming, "Error detecting and error correcting codes," *Bell Syst. Tech. J.*, vol. 29, no. 2, pp. 147–160, 1950.
[40] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "DDoS in the IoT: Mirai and other botnets," *Computer*, vol. 50, no. 7, pp. 80–84, 2017.
[41] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*. London, U.K.: Routledge, 2017.
[42] M. A. Kramer, "Nonlinear principal component analysis using autoassociative neural networks," *AIChE J.*, vol. 37, no. 2, pp. 233–243, 1991.
[43] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

Jiamin Fan received the B.S. degree in telecommunications engineering from Nanjing University of Posts and Telecommunications, Nanjing, China, in 2014, and the M.S. and Ph.D. degrees in computer science from the University of Victoria, Victoria, BC, Canada, in 2018 and 2023, respectively.

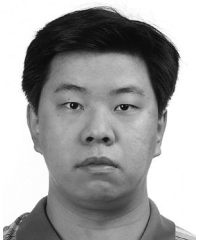
Her research interests include networking, machine learning, and distributed computing.

Guoming Tang (Member, IEEE) received the bachelor's and master's degrees from the National University of Defense Technology, Changsha, China, in 2010 and 2012, respectively, and the Ph.D. degree in computer science from the University of Victoria, Victoria, BC, Canada, in 2017.

He is a Research Fellow with Peng Cheng Laboratory, Shenzhen, Guangdong, China. He was also a Visiting Research Scholar with the University of Waterloo, Waterloo, ON, Canada. His research mainly focuses on green computing and cloud/edge computing.

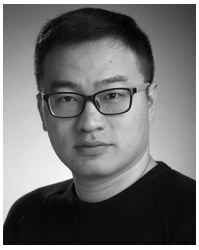
Kui Wu (Senior Member, IEEE) received the B.Sc. and M.Sc. degrees in computer science from Wuhan University, Wuhan, China, in 1990 and 1993, respectively, and the Ph.D. degree in computing science from the University of Alberta, Edmonton, AB, Canada, in 2002.

He joined the Department of Computer Science, University of Victoria, Victoria, BC, Canada, in 2002, where he is currently a Full Professor. His current research interests include network performance analysis, online social networks, Internet of Things, and parallel and distributed algorithms.



Zhengan Zhao (Graduate Student Member, IEEE) received the B.Sc. degree in computer science from Miami University, Oxford, OH, USA, in 2020, and the M.Sc. degree in computer science from the University of Victoria, Victoria, BC, Canada, in 2022, where he is currently pursuing the Ph.D. degree with the Department of Computer Science.

His research interests include IoT network anomaly detection and access control.



Yang Zhou received the B.S. degree in electronic engineering from Jilin University, Changchun, China, in 2011, the M.S. degree in electronic engineering from Peking University, Beijing, China, in 2014, and the M.S. degree in computer science from Simon Fraser University, Burnaby, Canada, in 2017.

He is currently a Senior Data Scientist and a Cyber Security Researcher. His main research interests include malware detection and classification with machine learning techniques and attacks against operation systems and networks.

Shengqiang Huang received the Bachelor of Engineering degree from Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 1999.

He is currently a Senior Principal Engineer with Huawei Technologies Canada Company Ltd., Vancouver, BC, Canada. He has more than 20 years of extensive experience in the network security industry, reverse engineering, and general software architecture with roles that span across organization and business maturity. He is leading the antivirus and sandbox product research with Huawei. Before Huawei, he held various product management and development roles with GE and Fortinet. His specialties are antivirus engine development, intrusion prevention systems, sandboxing, industrial control systems security, APT/malware, NGFW, application layer security, and UTM.