Federated Anomaly Detection on System Logs for the Internet of Things: A Customizable and Communication-Efficient Approach

Beibei Li, *Member, IEEE*, Shang Ma, *Student Member, IEEE*, Ruilong Deng, *Senior Member, IEEE*, Kim-Kwang Raymond Choo, *Senior Member, IEEE*, and Jin Yang

***Abstract*—Runtime log-based anomaly detection is one of several key building blocks in ensuring system security, as well as post-incident forensic investigations. However, existing log-based anomaly detection approaches that are implemented on large-scale Internet of Things (IoT) systems generally upload local data from edge devices to a centralized (cloud) server for processing and analysis. Such a workflow incurs significant communication and computation overheads, with potential privacy implications. Hence, in this paper, we propose a customizable and communication-efficient federated anomaly detection scheme (hereafter referred to as FedLog), designed to facilitate the identification of abnormal log patterns in large-scale IoT systems. Specifically, we first craft a Temporal Convolutional Network-Attention Mechanism-based Convolutional Neural Network (TCN-ACNN) model, to effectively extract fine-grained features from system logs. Second, we develop a new federated learning framework to support IoT devices in establishing a comprehensive anomaly detection model in a collaborative and privacy-preserving manner. Third, a lottery ticket hypothesis based masking strategy is designed to achieve customizable and communication-efficient federated learning in handling non-Independent and Identically Distributed (non-IID) log datasets. We then evaluate the performance of our proposed scheme with those of DeepLog (published in CCS, 2017) and Loganomaly (published in IJCAI, 2019) in both centralized learning and federated learning settings, using two publicly available and widely used real-world datasets (i.e., HDFS and BGL). The findings demonstrate the utility of the proposed FedLog scheme, in terms of log-based anomaly detection.**

***Index Terms*—Log analysis, Internet of Things (IoT), federated learning, data privacy, artificial intelligence.**

Manuscript received August 20, 2021; revised November 25, 2021 and February 10, 2022; accepted February 10, 2022. Date of publication February 18, 2022; date of current version June 10, 2022. This work was supported in part by the National Key Research and Development Program of China under Grant No. 2020YFB1805400; the National Natural Science Foundation of China under Grants No. 62002248, No. U19A2068, No. 62073285, No. 62061130220, No. 61872254, and No. 62162057, and in part by the Natural Science Foundation of Zhejiang Province under Grant LZ21F020006. The work of Kim-Kwang Raymond Choo was supported only by the Cloud Technology Endowed Professorship. The associate editor coordinating the review of this article and approving it for publication was M. Alazab. *(Corresponding author: Jin Yang.)*

Beibei Li, Shang Ma, and Jin Yang are with the School of Cyber Science and Engineering, Sichuan University, Chengdu 610065, China (e-mail: libeibei@scu.edu.cn; mashang@stu.scu.edu.cn; yangjin@scu.edu.cn).

Ruilong Deng is with the State Key Laboratory of Industrial Control Technology and the College of Control Science and Engineering, Zhejiang University, Hangzhou 310027, China (e-mail: dengruilong@zju.edu.cn).

Kim-Kwang Raymond Choo is with the Department of Information Systems and Cyber Security, University of Texas at San Antonio, San Antonio, TX 78249 USA (e-mail: raymond.choo@fulbrightmail.org).

Digital Object Identifier 10.1109/TNSM.2022.3152620

I. Introduction

Modern large-scale Internet of Things (IoT) systems generally comprise hundreds to (tens of) thousands of edge devices, which may differ significantly in terms of technical specifications (e.g., their design goals, computational and communication capabilities, battery life). However, given the sensitivity of the data sensed, collected, sent, and received by these edge devices, we have to ensure that the system/ecosystem is secure. One of the existing approaches relies on the recording and analysis of system logs [1]–[4]. The capability to accurately detect runtime anomalies enables system operators and organizations to effectively identify/detect, correlate, and respond to faults and cyberthreats in a timely fashion. However, as existing systems become more interconnected and the number of connected devices significantly increases, the volume of log data will also expand correspondingly. This compounds the challenge of real-time (or just-in-time) automated log inspection.

There have been attempts to utilize Artificial Intelligence (AI), broadly defined to include both machine and deep learning, in anomaly detection [3], [5], [6], and example approaches include those presented in [7]–[12]. While machine/deep-learning based approaches can facilitate the understanding and represent of unstructured free-text log data, existing approaches are generally designed for standalone systems and are less effective in large-scale IoT systems (with a massive number of edge devices generating a huge amount of log data). It is also generally acknowledged that a single edge device in a large ecosystem is unlikely to generate sufficient data to support the training of a machine/deep learning model. This, perhaps, explains why collaborative learning (that allows one to more fully utilize distributed intelligence from multiple devices and systems to provide global situational awareness) is increasingly popular. However, most existing schemes are mainly based on a centralized learning design (see Fig. 1); that is, a centralized node (e.g., cloud server) collects system logs from distributed devices (e.g., edge devices), and trains an anomaly model on the collected data. There are, however, limitations in a centralized learning design paradigm. First, with ever-increasing log data sizes, the centralized server incurs significant computation and communication overheads [2]. Second, system logs contain sensitive information about IoT/edge devices [13], and there are potential privacy

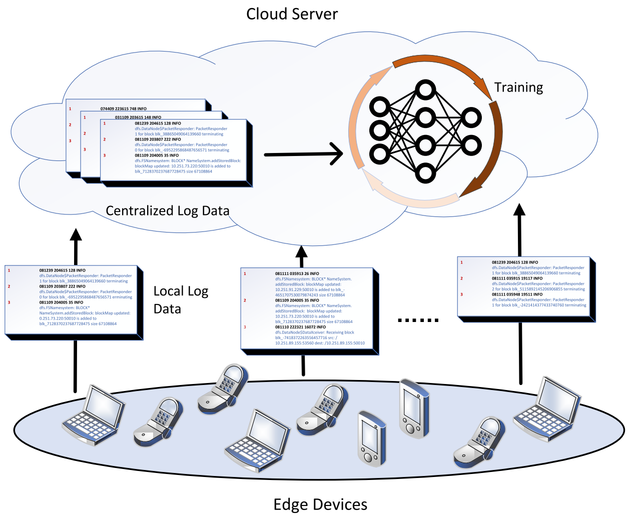


Fig. 1. Conventional centralized log-based anomaly detection.

risks in sending all log data to a centralized server, particularly if the server is in a different jurisdiction.

Federated learning extends collaborative learning by allowing local clients to cooperate and train a global model, without the need for participants to share their data. Such an approach can potentially facilitate privacy protection in AI-enabled IoT systems [14], [15]. For example, in 2019, Nguyen *et al.* [14] proposed an anomaly detection-based intrusion detection system that adopts a federated Gated Recurrent Units (GRU) network to detect anomalies in IoT systems. In 2020, Liu *et al.* [15] presented a federated Attention Mechanism-based Convolutional Neural Network-Long Short Term Memory model to facilitate anomaly detection in Industrial IoT systems.

There are a number of ongoing challenges in applying federated learning to log-based anomaly detection. First, federated learning tasks are typically performed using deep learning models, with fast evolving parameters. Directly transferring the model without any modification to the underlying structure could produce significant communication overheads [16]. Second, non-Independent and Identically Distributed (non-IID) data is common in real-world systems where each client generates diverse data based on the specific job. This usually results in increased statistical heterogeneity across clients, which could complicate the convergence of the global model [17]. Furthermore, customizable anomaly detection is required to accommodate the non-IID data across clients [18]. Seeking to address the above-discussed challenges, we present a federated anomaly detection scheme (FedLog). The latter is designed to facilitate accurate and efficient anomaly detection on log data from large-scale IoT systems. FedLog comprises the following key building blocks.

- We craft an unsupervised deep anomaly detection model (coined TCN-ACNN), to effectively learn the temporal representation and detect the anomalies in system logs.
- We develop a new federated learning framework, to support IoT devices in establishing a comprehensive anomaly detection model in a collaborative and privacy-preserving manner.
- We design a lottery ticket hypothesis [19] based masking strategy, to significantly reduce communication overheads and enhance the customization of federated learning in handling non-IID datasets.

The rest of this paper is organized as follows. In the next two sections, we will review the extant literature and present the preliminaries required in the understanding of our proposed scheme. In the fourth section, we will introduce our system model and the challenges. We present our proposed scheme and its performance evaluation in the fifth and sixth sections, prior to concluding this work in the last section.

II. Other Existing Approaches

In this section, we briefly review the other competing approaches for both log-based anomaly detection and federated learning.

A. Log-Based Anomaly Detection

Detecting anomalies on log data using deep learning has been actively studied in recent years. For example, in 2017, Du *et al.* [7] demonstrated a Long Short-Term Memory (LSTM)-based approach to detect sequential and quantitive anomalies from log key sequences and parameter values. In the same year, He *et al.* [20] revealed the inefficiency of current log parsers while dealing with large-scale log data and constructed a novel log parser capable of processing large-scale log data in parallel systems. In 2019, Meng *et al.* [8] proposed a novel log template representation method by incorporating semantic information from log templates and presented an attention mechanism-based LSTM model to detect anomalies. In the same year, Zhang *et al.* [9] addressed the instability of log data by proposing a robust anomaly detector that extracts the semantic information of log sequences. In late 2021, Huang *et al.* [10] designed a hierarchical transformer that encodes log template sequences and parameter values into high-level representation and built an attention mechanism-based classifier to detect anomalies.

B. Challenges to Federated Learning

Recent years have witnessed a growing research interest in federated learning. However, several challenges are remaining unsolved, e.g., expensive communication, statistical heterogeneity. Expensive communication is a major challenge for the application of federated learning. The primary method of studies in improving communication efficiency is to modify the client model's updates. For example, in 2016, Konečnỳ *et al.* [21] proposed two approaches, structured updates and sketched updates, to reduce communication costs in federated learning. In 2020, Liu *et al.* [15] presented a gradient compression mechanism by keeping the gradient with a larger absolute value only. To tackle statistical heterogeneity, i.e., non-IID data, researchers target at either deriving a well-converged global model [22] or building customizable local models [23]. For example, Zhao *et al.* [22] in 2019 proposed a data-sharing mechanism to lessen the influence of the non-IID data on the global model. In 2020, Shen *et al.* [23] combined deep mutual learning [24] with federated learning by maintaining an additional local model for each client, which exchanges knowledge with the model learned by federated learning.

Moreover, many efforts [16], [18], [25], [26] have been devoted to addressing expensive communication and statistical heterogeneity at the same time. For instance, in 2019, Sattler *et al.* [16] designed sparse ternary compression (STC), to cope with these two problems. In 2020, both Diao *et al.* [26] and Liang *et al.* [25] proposed to improve communication efficiency of federated learning on non-IID datasets by sharing part of the model parameters. However, the performance of the local model is ignored in the study of Diao *et al.* while no global model can be directly obtained in the study of Liang *et al.* In contrast, the proposed framework produces customizable models for both the server and clients. Li *et al.* [18] also combined the lottery ticket hypothesis with federated learning. However, their work requires training and transferring the mask in all communication rounds for each client. Instead, only one-time mask training and transferring is demanded for each client in our scheme. In comparison, the proposed framework only requires training and delivering the mask to the server only once for each client, which significantly reduces computation and communication costs.

III. Preliminaries

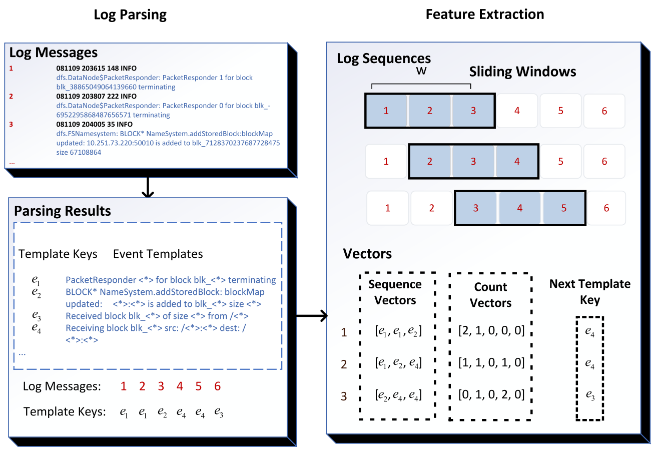
In this section, we briefly introduce some preliminaries of log-based anomaly detection and federated learning.

A. Log-Based Anomaly Detection

In addition to log collection, the overall workflow of log-based anomaly detection comprises three parts: log parsing, feature extraction, and anomaly detection [27]. In this part, we illustrate them with an example of HDFS (Hadoop Distributed File System) logs [1].

1) Log Parsing: Log messages are unstructured texts consisting of two parts, i.e., a constant part named event template and a varying part named log parameters. The objective of log parsing is to extract the event template and log parameters from each log message and create a dictionary mapping each event template to a unique template key. In this way, the unstructured log messages are converted into structured data. Figure 2 displays an example of parsing HDFS logs. For instance, the log message *"081109 203615 148 INFO dfs.DataNode$PacketResponder: PacketResponder 1 for block blk_38865049064139660 terminating"* records the event *"PacketResponder 1 for block blk_38865049064139660 terminating"* where the words *"PacketResponder"*, *"for"*, *"block"*, *"blk_"*, *"terminating"* are constant event template and are extracted as *"PacketResponder $< * >$ for block blk_$< * >$ terminating"*. Note that $< * >$ stands for the log parameters, i.e., *"1"* and *"38865049064139660"*.

2) Feature Extraction: The objective of feature extraction is to convert structured log data into numerical vectors. We first slice log data into a collection of sliding windows. The log messages that fall on the same window constitute a log sequence. As shown in Fig. 2, if the window size is set as 3 and the step size is set as 1, we can obtain three log sequences, i.e., [1, 2, 3], [2, 3, 4], and [3, 4, 5], from the log message series [1, 2, 3, 4, 5]. Further, we compute two vectors based on each derived log sequence, i.e., a sequence vector represents the sequence of the template keys in the log sequence and a count vector represents the number of occurrences of each template key in the log sequence. Besides, we record the next template key of each window as label to train the anomaly detection model. Specifically, given a log sequence, the sequence vector is formulated as



```
1   081109 203615 148 INFO
    dfs.DataNode$PacketResponder: PacketResponder 1 for block
    blk_38865049064139660 terminating
2   081109 203807 222 INFO
    dfs.DataNode$PacketResponder: PacketResponder 0 for block blk_-
    6952295868487656571 terminating
3   081109 204005 35 INFO
    dfs.FSNamesystem: BLOCK* NameSystem.addStoredBlock:blockMap
    updated: 10.251.73.220:50010 is added to blk_7128370237687728475
    size 67108864
...
```

```
e1   PacketResponder <*> for block blk_<*> terminating
e2   BLOCK* NameSystem.addStoredBlock: blockMap
     updated:    <*>:<*> is added to blk_<*> size <*>
e3   Received block blk_<*> of size <*> from /<*>
e4   Receiving block blk_<*> src: /<*>:<*> dest: /
     <*>:<*>
...
```

Log Messages: 1 2 3 4 5 6
Template Keys: e_1 e_1 e_2 e_4 e_4 e_3
Log Sequences
W
Sliding Windows
1 2 3 4 5 6
1 2 3 4 5 6
1 2 3 4 5 6
Vectors
Sequence Vectors
Count Vectors
Next Template Key
1 $[e_1, e_1, e_2]$ [2, 1, 0, 0, 0] e_4
2 $[e_1, e_2, e_4]$ [1, 1, 0, 1, 0] e_4
3 $[e_2, e_4, e_4]$ [0, 1, 0, 2, 0] e_3

Fig. 2. An illustration of log parsing and feature extraction.

$$s_t = [m_{t-h}, m_{t-h+1}, \ldots, m_{t-1}], \tag{1}$$

where m_i denotes the value of the template key at timestamp *i* and *h* is the window size. Let $E_n = \{e_1, e_2, \ldots, e_n\}$ denote the set of *n* distinct template keys in the log dataset (clearly, $m_i \in E_n$). The count vector is formulated as

$$p_t = [count(e_1), count(e_2), \ldots, count(e_n)], \tag{2}$$

where $count(e_i)$ denotes the number of occurrences of each template key in the log sequence. For instance, as illustrated in Fig. 2, given the log sequence [1, 2, 3], we can obtain $s_t = [e_1, e_1, e_2]$ and $p_t = [2, 1, 0, 0, 0]$ when $n = 5$.

3) Anomaly Detection: Anomaly detection focuses on identifying data patterns that have significant differences from normal observations. It has been widely applied to areas like network security management and performance assurance [28]. There are mainly three categories of anomalies [29].

- *Point Anomalies:* Point anomalies are individual data instances that are considered abnormal with respect to the rest of the data.
- *Contextual Anomalies:* Contextual anomalies refer to the individual data instances that are considered abnormal in a specific context.
- *Collective Anomalies:* Collective anomalies are a group of related data instances that are abnormal compared to the rest of the data.

The objective of log-based anomaly detection is to uncover the collective anomalies, which are abnormal log events that appeared in sequences.

Anomaly detection tasks can be performed with both supervised and unsupervised learning. In this work, we focus on unsupervised anomaly detection because manual labeling the exploding amount of log data is extremely expensive and impractical. To this end, instead of directly classifying labels (i.e., abnormal or normal) of each log sequence, we predict log messages with respect to the historical log sequences. Specifically, if the template key of the incoming log message differs from the predicted one, the log sequence which contains that log message is considered abnormal. More details can be referred to Section V.

B. Federated Learning

Recent years have witnessed a growing research interest in federated learning [14], [15], [30], [31]. Federated learning enables training a global machine learning model on decentralized datasets from clients without sharing the training data. The most widely adopted federated learning algorithm is Federated Averaging (FedAvg) [32]. Formally, in FedAvg, the cloud server first initializes the global model and then coordinates an iterative training procedure with three steps involved in each round, which are described as follows.

- *Broadcasting:* The server selects a random set of M clients to participate in the round train and broadcasts the global model to them.
- *Local Training:* Each client trains the received model by performing Stochastic Gradient Descent (SGD) [33] on the local dataset and sends the model updates back to the cloud server.
- *Model Aggregating:* The cloud server averages the received model updates as the global model by

$$\theta^{t+1} = \sum_{k=1}^{M} \frac{1}{M} \theta_k^t, \tag{3}$$

where θ_k^t denotes the model update from client k in t-th communication round and θ^{t+1} is the global model in $(t + 1)$-th communication round.

IV. System Model and Challenges

In this section, we introduce the system model and challenges considered in this work. Then, we also discuss the design goals of the proposed scheme.

A. System Model

The system model under consideration is a federated learning framework for log-based anomaly detection in large-scale IoT systems (see Fig. 3), where three entities are involved.

- *Cloud Server:* The cloud server is equipped with massive computing and storage resources to support big data processing. It coordinates the federated learning task by aggregating the model updates and broadcasting the global model, without accessing the training data.

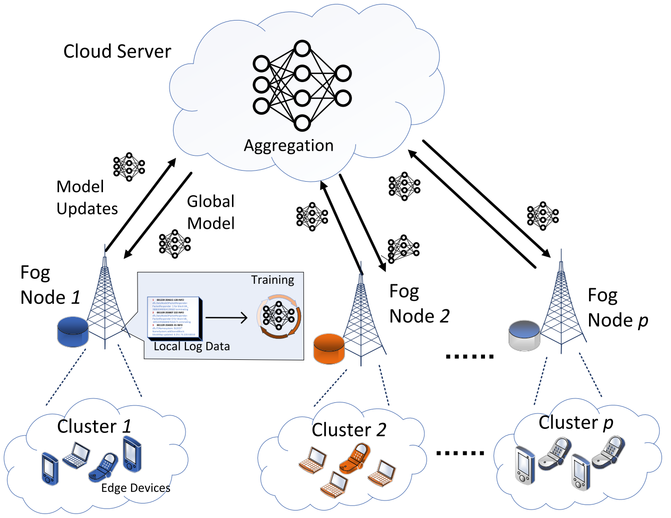


Fig. 3. The system model under consideration.

- *Fog Nodes:* Fog nodes are devices endowed with sufficient capacities to support IoT applications. Each fog node is responsible for storing and analyzing the system logs collected from a cluster of edge devices sharing similar geographic locations, and building a federated anomaly detection model together with the cloud server.
- *Edge Devices:* Edge devices monitor and collect runtime events as system logs while handling the routine work. Since edge devices are usually resource-constrained, system logs are delivered to the fog node for further storage and analysis, such as anomaly detection.

B. Challenges

Though federated learning is capable of improving scalability and preserving data privacy, performing federated learning directly with conventional log-based anomaly detection approaches still faces three challenges (as shown below).

1) Heavy Training: Many conventional log-based anomaly detection approaches are based on Recurrent Neural Networks (RNNs), which usually demand lots of memory and time to train and not support parallel training [34]. Consequently, the iterative training of federated learning could occupy enormous resources and impede the fog node from routine works.

2) Expensive Communication: Deep learning models are architected to become increasingly heavier with millions of parameters in order to boost the capacity in modern applications [16]. Although the transferring of training data between the server and each client is avoided, traditional federated learning frameworks (e.g., FedAvg) may still produce huge communication overheads by constantly exchanging the deep learning model.

3) Heterogeneous Datasets: Edge devices in large-scale IoT systems generate diverse heterogeneous datasets according to their specific jobs in the system, which results in the non-IID data across clusters. Such data can usually affect the federated learning tasks in two aspects. On the one hand, it may complicate the convergence of the global model thus causing significant degradation in the model's performance [17]. On the other hand, a single unique global model can not accommodate the non-IID data to provide customizable anomaly detection to each cluster [26].

C. Design Goals

To address the above challenges, the key objective of the proposed scheme is to provide a customizable and communication-efficient federated learning framework for anomaly detection on system logs. Our design goals are given as follows.

- As we leverage federated learning for privacy-preserving log-based anomaly detection, we would like to demonstrate that the performance of federated learning based models should be at least comparative with or superior compared to traditional centralized learning based ones.
- To address the challenge of heavy training, we proposed an unsupervised anomaly detection model consisting of Temporal Convolutional Network (TCN), Convolutional Neural Network (CNN), and attention mechanism. It should be able to match or exceed the performance of the state-of-the-art models that rely on RNNs.
- To accommodate the challenges of expensive communication and heterogeneous datasets, we develop a new federated learning framework by incorporating the lottery ticket hypothesis. This framework should improve the communication efficiency of the proposed scheme under both IID and non-IID scenarios and increase its accuracy under the non-IID scenario.

V. The Proposed FedLog Scheme

In this section, we elaborate on the proposed scheme by first introducing the TCN-ACNN based anomaly detection model, followed by detailing the customizable and communication-efficient federated learning framework.

A. The TCN-ACNN-Based Anomaly Detection Model

In this part, we introduce the architecture of the TCN-ACNN model as well as the algorithm of anomaly detection.

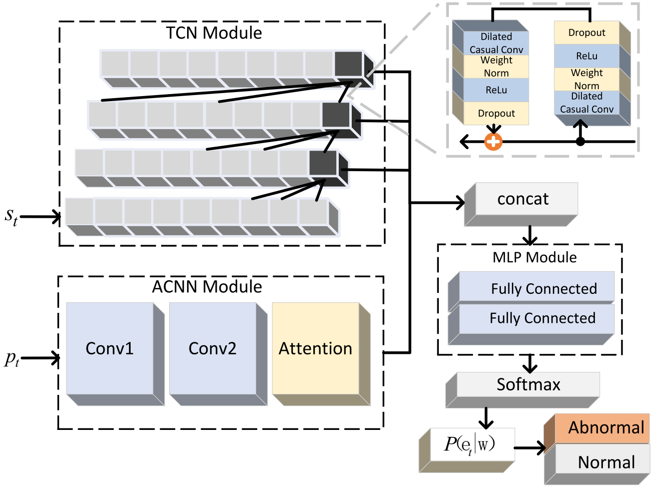


Fig. 4. The architecture of designed TCN-ACNN model.

1) Model Architecture: The designed TCN-ACNN is composed of a TCN module, a CNN module, followed by an MLP module, and a softmax layer, which is shown in Fig. 4. Specifically, the TCN module is a TCN with two hidden layers and an output layer. TCN follows the architecture of a 1D fully-convolutional network, having the same input and output length. Moreover, it leverages three components to achieve better capacity in sequence modeling [34]. First, TCN utilizes casual convolutions which prevent any information leakage from the future to the past. Second, TCN incorporates a dilated convolution mechanism that allows it to see far deeper into the past instead of being restricted in a linear receptive field. Formally, the dilated convolution operation *F* applied on the sequence element x_t with respect to a filter $f : \{0, \ldots, k-1\} \rightarrow \mathbb{R}$ is given by

$$F(x_t) = \sum_{i=0}^{k-1} f(i) \cdot x_{t-di}, \tag{4}$$

where *k* is the kernel size of the filter and *d* is the dilation factor. We set $d = 2^i$ for the *i*-th layer of the TCN. Third, TCN employs residual connections to avoid the problem of vanishing and exploding gradients. A general form of residual connection is given by

$$output = Activation(input + F(input)), \tag{5}$$

where *F* is the operation applied to the input. We adopt ReLU as the activation function in our design. In addition, we concatenates the units of two hidden layers and the output layer at timestamp *t* as the output of TCN module.

The ACNN module is composed of two convolutional blocks followed by an attention layer. Each convolution block involves a 1D convolutional layer and a max-pooling layer. We compute the attention score by

$$r_t = v^T tanh(Wh_t), \tag{6}$$

where r_t is the attention score corresponding to the hidden state h_t at timestamp *t*, while *W* and *v* are weight variables. We then apply softmax on r_t as $\alpha_t = Softmax(r_t)$, and multiply α_t with h_t as the final output of the attention layer.

2) Anomaly Detection: We perform the unsupervised anomaly detection based on time-series forecasting. Specifically, we first input the sequence vector s_t and the count vector p_t to the TCN and ACNN module, respectively, to learn hidden representation. Detailed procedures of deriving s_t and p_t from log messages can be found in Section III. Then, we concatenate the outputs of the TCN module and the ACNN module and feed them to the MLP module, which is composed of two fully connected layers. The overall process can be described by

$$\begin{aligned} h_1 &= TCN(s_t) \\ h_2 &= ACNN(p_t) \\ \tau &= MLP(h_1; h_2), \end{aligned} \tag{7}$$

where h_1, h_2 are hidden vectors and τ denotes the output of the MLP module. τ is then forwarded to the softmax layer in purpose of, given the past window w, obtaining the probability distribution of m_t, which is calculated by

$$P(m_t = e_i|w) = Softmax(\tau). \quad (8)$$

At training time, we calculate the cross-entropy loss of $P(m_t = e_i|w)$ and the label derived in the feature extraction part. Further, we use SGD to train the parameters of the TCN-ACNN model. At inference time, we sort $P(m_t = e_i|w)$ by e_i and choose the top c template keys with the highest probability as event candidates, which is denoted as E_c. If the template key of the incoming log message is not one of these candidates, i.e., $m_t \notin E_c$, the log sequence which contains that log message is classified abnormal.

B. The Customizable and Communication-Efficient Federated Learning Framework

In federated learning, several works [25], [26] also present that clients are able to share part of the local models while comparable or even better performance can be reached compared to FedAvg. Pruning approaches could remove the unnecessary structure from neural networks so that federated learning can be more efficiently conducted by sharing pruned models. Instead of being developed for efficient inference computation and storage [35], [36], the lottery ticket hypothesis [19] shows the effectiveness of pruning at initialization. Therefore, we consider the scenario of using the lottery ticket hypothesis for federated learning

The lottery ticket hypothesis demonstrates an approach to extract such sparse subnetworks from the base network that reach comparable or even exceeding test accuracy through an iterative training, pruning, and initialization process [19]. Formally, the subnetworks, called the Lottery Ticket Networks (LTNs), are given as $m \odot \theta$, where $m \in \{0,1\}^{|\theta|}$ is the mask on the base network θ and $\odot$ computes element-wise product.

Each client, i.e., fog node, k with its individual dataset D_k obtains a particular LTN θ_k with a corresponding mask m_k. Our goal is to seek such an LTN for each client. Then, by performing federated learning on customized LTNs, we can adapt the global model to the non-IID datasets with unnecessary information excluded. Moreover, we can also achieve better customization by maintaining a customizable LTN for each client. Simultaneously, the sharing process is communication-efficient, because only the parameters of LTNs are exchanged between the server and each client.

Therefore, we present a masking strategy by exploiting the lottery ticket hypothesis to build our federated learning framework. The proposed federated learning framework consists of two phases, i.e., local mask training and masked federated learning, which are detailed as follows.

1) Local Mask Training: In this phase, all clients locally seek their LTNs and the corresponding masks using an iterative pruning method (see in Algorithm 1). Then they transmit the obtained masks to the cloud server. Specifically, each client k first initializes the local model and the mask.

```
Algorithm 1: Local Mask Training
Input: The number of pruning iteration $I_p$, the local
       training set $D_k^{train}$, the local validation set $D_k^{val}$,
       the total pruning rate $r_p$, the validation threshold
       $Acc_{thr}$, $D_k$, $K$, $E$, $B$, $\eta$.
for each client $k = 1, 2 \dots K$ do
    Initialize $\theta_k^1$, $m_k^1 = \{1\}^{|\theta_k^1|}$;
    for each pruning iteration i from $1, 2, \dots, I_p$ do
        $\mathcal{B} \leftarrow$ split local training data $D_k^{train}$ into batches
        of size $B$;
        for each local epoch $e = 1, 2, \dots, E$ do
            for batch $b \in \mathcal{B}$ do
                $\theta_k^i = \theta_k^i - \eta \nabla_{\theta_k^i} \ell(\theta_k^i; b)$;
            end
            $Acc_{val} \leftarrow$ evaluate $\theta_k^i$ on $D_k^{val}$;
            if $Acc_{val} > Acc_{thr}$ then
                break;
            end
        end
        $m_k^i \leftarrow$ prune $\theta_k^i$ with pruning rate $r_p \frac{i}{I_p}$ to get a
        new mask for the LTN;
        $\theta_k^{i+1} = \theta_k^i \odot m_k^i$ (reset the masked parameters
        $\theta_k^i \odot m_k^i$ to corresponding values in $\theta_k^1$ );
    end
    Transmit $m_k^{I_p}$ to server as $m_k$;
end
```

Then, during the *i*-th pruning iteration, the client k trains E epochs on the local dataset and prunes its model with a pruning rate of $r_p \frac{i}{I_p}$ to get the mask m_k^i, where r_p is the total pruning rate and I_p is the total number of pruning iterations. Note that after each training epoch, the client evaluates its model on the local validation set. If the validation accuracy is higher than the predefined threshold, the client stops training and directly prunes its model. The pruning method is detailed as follows.

- Ranking the unmasked weights in each layer.
- Setting the mask values of the top $r_p \frac{i}{I_p}$ weights with the smallest magnitude to 0 and the rest to 1.
- Reinitializing the weights with mask value 1. Setting the values of weights with mask value 0 to 0 and freezing them.

2) Masked Federated Learning: In this phase, the cloud server coordinates the federated learning based on the masks received from the clients in the previous phase. Importantly, on the server side, we create a new model aggregation algorithm (see Algorithm 2). Given *t*-th communication round, the cloud server first selects a random set of *M* fog nodes, denoted as S_t, to participate in the round training. For each client $k \in S_t$, the cloud server masks the global model using the corresponding mask to derive the LTN by $\theta_k^t = m_k \odot \theta^t$, and sends θ_k^t to client *k*. After training *E* epochs on the local training set, client *k* masks the client model by $\theta_k = m_k \odot \theta_k$, and transfers it to the cloud server.

Algorithm 2: Masked Federated Learning

```
Input: $D_k, K, E, B, \eta, m_1, m_2, \ldots, m_k$.
Server executes:
    Initialize global model $\theta^0$;
    for each round $t = 0, 1, \ldots$ do
        $S_t \leftarrow$ (random set of $M$ clients);
        for each client $k \in S_t$ in parallel do
            $\theta_k^t = m_k \odot \theta^t$;
            $\theta_k^{t+1} = ClientUpdate(k, \theta_k^t)$;
        end
        $\omega = \varphi\left(\sum_{k=1}^{M} m_k\right)$ ;
        $\theta^{t+1} = \omega \odot \sum_{k=1}^{M} \theta_k^{t+1}$;
    end

ClientUpdate$(k, \theta_k)$:
    $\mathcal{B} \leftarrow$ Split local data $D_k^{\text{train}}$ into batches of size $B$;
    $\theta_k \leftarrow$ Freeze the weights with mask value 0;
    for each local epoch $e = 1, 2, \ldots, E$ do
        for batch $b \in \mathcal{B}$ do
            $\theta_k = \theta_k - \eta \nabla \ell(\theta_k; b)$;
        end
    end
    Return $\theta_k$ to server;
```

Considering that each masked client model only contributes to part of the global model's weight parameters, the server can not simply average the received models by the number of participants M like Eq. (3) does. Therefore, we have to dig into the parameter level for aggregation. Specifically, we first count the number of contributors for each weight parameter to compute a weight mask ω, which given by

$$\omega = \varphi\left(\sum_{k=1}^{M} m_k\right), \tag{9}$$

where function φ maps each element x of the input matrix to its reciprocal. The mapping function is defined by

$$g(x) = \begin{cases} 0, & \text{if } x = 0. \\ \frac{1}{x}, & \text{otherwise.} \end{cases} \tag{10}$$

Note that a 0 element is mapped to 0 because the corresponding 0 weight is masked and contributes nothing in aggregation. Then, we divide each weight parameter's value of the aggregated model by the number of contributors, which is given by

$$\theta^{t+1} = \omega \odot \sum_{k=1}^{M} \theta_k^{t+1}. \tag{11}$$

The overall procedure of the proposed masked federated learning can be seen in Fig. 5.

VI. Performance Evaluation

In this section, we carry out a series of experiments to evaluate the performance of the proposed scheme. First, we give a description of the experiment settings. Then, we compare the performance of the proposed TCN-ACNN with other competing log-based anomaly detection models [7], [8] under both centralized learning and federated learning paradigms. Furthermore, we also compare the performance of the proposed federated learning framework with baselines.

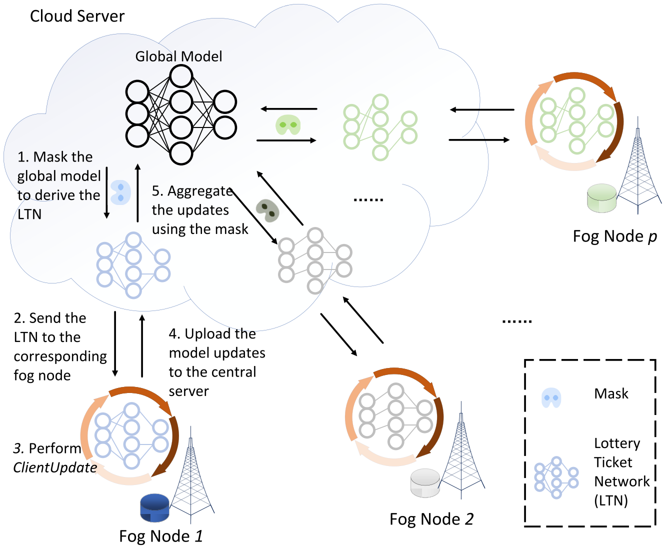

Cloud Server
Global Model
1. Mask the global model to derive the LTN
5. Aggregate the updates using the mask
......
Fog Node *p*
2. Send the LTN to the corresponding fog node
4. Upload the model updates to the central server
......
3. Perform *ClientUpdate*
Fog Node *1*
Fog Node *2*
Mask
Lottery Ticket Network (LTN)

Fig. 5. The proposed masked federated learning.

TABLE I
Log Datasets

Log datasets	Number of log sequences		Number of event templates
	Training set	Test set	
HDFS	46575 normal	157565 normal; 241873 abnormal	28
BGL	130259 normal	67940 normal; 68658 abnormal	312

A. Experiment Settings

1) Datasets: We evaluate the proposed scheme on two benchmark datasets derived in real-world systems, namely the HDFS dataset [1] and the BGL (Blue Gene/L) dataset [37]. Both of them are manually labeled and publicly available.[1] They have been widely employed for log analysis by researchers [1]–[3], [7]–[10]. A brief introduction to them is given as follows (see also a brief summary in Table I).

- *HDFS:* The HDFS dataset contains 11,175,629 log messages and is generated by performing Hadoop-based map-reduce tasks on more than 200 Amazon's EC2 nodes.
- *BGL:* The BGL dataset consists of 4,747,963 logs messages. It is produced by the Blue Gene/L supercomputer at Lawrence Livermore National Laboratory (LLNL).

We synthesize the IID data in federated learning by randomly allocating each client an equal-size dataset with a balanced distribution of all classes (i.e., the next template

[1] https://github.com/logpai/loghub

TABLE II
PARAMETERS OF THE PROPOSED FEDERATED LEARNING FRAMEWORK

Phase	Description	Variable	Setting
Local mask training	No. clients	K	10
	No. local epochs	E	20
	Learning rate	η	0.001
	Minibatch size	B	64
	No. pruning iteration	I_p	4
	Validation threshold	Acc_thr	50
Masked federated learning	No. clients	K	10
	No. local epochs	E	1
	Learning rate	η	0.001
	Minibatch size	B	64
	No. participants in each round	M	10

key of each sliding window). Real-world partitioned datasets usually exhibit various types of non-IIDness (e.g., feature distribution skew, label distribution skew, etc.) [38]. In our case, edge devices generate diverse log sequence though they are all normal. Hence, we focus on the feature distribution skew type, where features of data vary across clients but the corresponding label may be the same. Therefore, we synthesize the non-IID data by randomly allocating each client an equal-size dataset consisting of solely two classes.

2) Environmental Setup and Metrics: We perform our evaluations on a laptop with Intel Core i7-9750F CPU 2.60GHz and an NVIDIA GeForce GTX 1650 GPU, and the deep learning models are implemented using PyTorch.[2] For log-based anomaly detection, we set window size $h = 10$ and the number of candidates $c = 9$. The default parameters of the proposed federated learning framework are shown in Table II.

Determining the abnormality of log sequences is a binary classification task. To this end, we employ four metrics for evaluation, namely 1) Accuracy: The correct proportion of the classification results. 2) Precision: The proportion of log sequences classified as anomalies that are indeed anomalies 3) Recall: The proportion of abnormal log sequences that are classified as anomalies 4) F1-score: The weighted average of precision and recall. Besides, we count the total volume of parameters exchanged between clients and the server to quantitatively evaluate communication overheads. To simplify our expression, we adopt the number of parameters of a single deep learning model, i.e., TCN-ACNN, as the unit. Given the number of communication rounds r, the number of participants in each round M, the sharing ratio R, we can calculate the number of communicated parameters C of federated learning using the following equation:

$$C = 2rMR. \tag{12}$$

For the proposed framework, $R = 1 - p_w r_p$, where $p_w = 0.63$, indicating the proportion of parameters in weights to those in total, and r_p denotes the pruning rate.

B. Performance Comparative Summary

We first compare the performance of TCN-ACNN, DeepLog [7], and Loganomaly [8] in both centralized learning and federated learning paradigm under the IID scenario. Specifically, as for federated learning, we implement DeepLog, Loganomaly with FedAvg, and TCN-ACNN with both FedAvg and FedLog ($r_p = 0.9$). To present more clearly, we denote the centralized learning paradigm of a log-based anomaly detection model with the subscript C, the FedAvg paradigm of a log-based anomaly detection model with the subscript F. For example, the centralized learning paradigm of TCN-ACNN is denoted as TCN-ACNN$_C$ while the FedAvg paradigm is denoted as TCN-ACNN$_F$. We only implement the FedLog paradigm of TCN-ACNN, denoted as TCN-ACNN$_{FedLog}$, because the lottery ticket hypothesis fails to work on DeepLog or Loganomaly.

TABLE III
PERFORMANCE COMPARISON UNDER IID AND NON-IID SCENARIOS

Approaches	R	Accuracy (%)		
		IID	Non-IID	
		Global	Local	Global
Standalone	0.00	99.63	66.84	53.45
FedAvg	1.00	99.48	25.01	99.48
LG-FedAvg	0.85	99.48	28.12	99.48
FedLog(0.4)	0.62	**99.63**	59.30	**99.63**
FedLog(0.3)	0.56	**99.63**	57.30	99.48
FedLog(0.2)	0.50	**99.63**	**63.30**	99.48
FedLog(0.1)	0.43	99.48	53.73	99.48
FedLog(0.05)	0.40	99.48	55.30	75.90
FedLog(0.01)	0.38	99.48	59.30	71.55

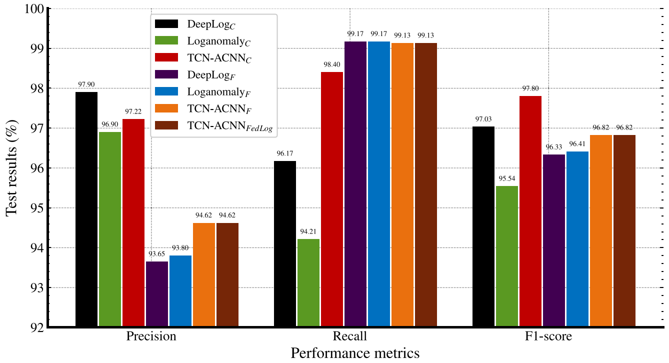


Fig. 6. Performance comparison on the HDFS dataset.

Figures 6 and 7 show the numerical results about the precision, recall, F1-score of different log-based anomaly detection models evaluated on the HDFS dataset and BGL dataset, respectively. For one thing, we can easily see that the proposed TCN-ACNN model achieves the best F1-score (97.80% of TCN-ACNN$_C$, 96.82% of both TCN-ACNN$_F$ and TCN-ACNN$_{FedLog}$) compared to DeepLog (97.03% of DeepLog$_C$ and 96.33% of DeepLog$_F$) and Loganomaly (95.54% of Loganomaly$_C$ and 96.41% of Loganomaly$_F$) on the HDFS dataset in terms of both centralized learning and federated learning paradigms. Additionally, TCN-ACNN also outperforms DeepLog and Loganomaly on the BGL dataset in terms of both paradigms with the best precision, recall, and F1-score (94.38%, 97.92%, 96.12% of TCN-ACNN$_C$

[2] Pytorch: The Python deep learning library https://pytorch.org/.

TABLE IV
COMPARISON OF CUSTOMIZATION AND COMMUNICATION OVERHEADS WITH VARYING ROUNDS UNDER NON-IID SCENARIO

Frameworks	R	$r=5$			$r=10$			$r=15$			$r=20$			$r=25$		
		C	Accuracy (%)		C	Accuracy (%)		C	Accuracy (%)		C	Accuracy (%)		C	Accuracy (%)	
			Local	Global		Local	Global		Local	Global		Local	Global		Local	Global
Standalone	0.00	0	28.37	52.85	0	41.84	52.96	0	48.23	53.04	0	55.20	53.07	0	57.56	53.07
FedAvg	1.00	100	21.38	85.61	200	25.01	99.48	300	25.01	99.48	400	25.01	99.48	500	25.01	99.48
LG-FedAvg	0.85	85	18.36	72.23	170	28.12	99.48	256	28.12	99.48	241	28.12	99.48	426	28.12	99.48
FedLog(0.4)	0.62	62	21.63	**85.80**	124	40.70	**99.63**	186	51.81	**99.63**	248	**55.11**	**99.63**	310	**55.11**	**99.63**
FedLog(0.3)	0.56	56	**25.50**	65.48	112	34.38	82.06	168	45.85	85.80	224	50.97	85.80	280	50.97	85.80
FedLog(0.2)	0.50	50	21.01	71.89	100	35.14	85.61	150	38.38	85.61	200	45.65	85.61	250	52.33	85.61
FedLog(0.1)	0.43	43	24.26	85.71	86	**41.08**	85.71	129	**51.89**	85.71	172	52.64	99.48	215	52.64	99.48
FedLog(0.05)	0.40	40	16.82	63.04	80	37.13	63.04	120	44.54	75.90	160	47.30	75.90	200	52.30	75.90
FedLog(0.01)	0.38	38	19.74	56.78	76	33.34	71.56	114	44.25	71.56	152	50.30	71.56	190	54.30	71.56

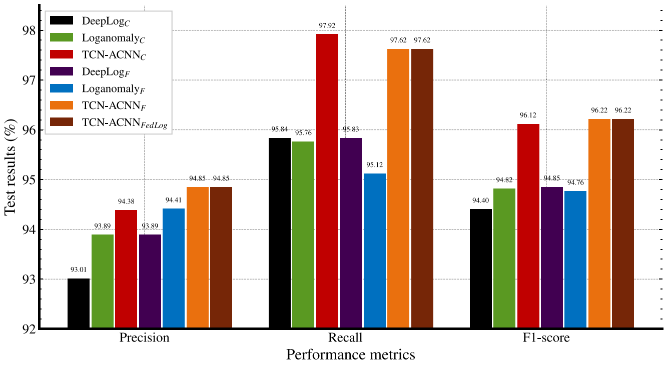


Fig. 7. Performance comparison on the BGL dataset.

and 94.85%, 97.62%, 96.22% of TCN-ACNN$_F$ and TCN-ACNN$_{FedLog}$). For another, we can observe that the federated learning paradigms of all the log-based anomaly detection models have comparable performance with the centralized learning paradigms in terms of F1-score. In particular, for the HDFS dataset, DeepLog$_F$ and TCN-ACNN$_F$ have a loss of 0.70% and 0.98% respectively compared to DeepLog$_C$ and TCN-ACNN$_C$ while Loganomaly$_F$ shows a 0.87% lead by Loganomaly$_C$. As for the BGL dataset, DeepLog$_F$ and TCN-ACNN$_F$ achieve a lead of 0.45%, 0.10%, respectively, while Loganomaly$_F$ has a negligible loss of 0.06%.

Therefore, we empirically demonstrate the superiority of the proposed TCN-ACNN and show the suitability of using federated learning for log-based anomaly detection. Notably, TCN-ACNN$_{FedLog}$ ($r_p = 0.9$) reaches the same performance as FedAvg, which is encouraging because it only uses 10% of the model's weight parameters. In the next part, we will further explore the effectiveness of the proposed federated learning framework, for which we adopt TCN-ACNN as the underlying deep learning model.

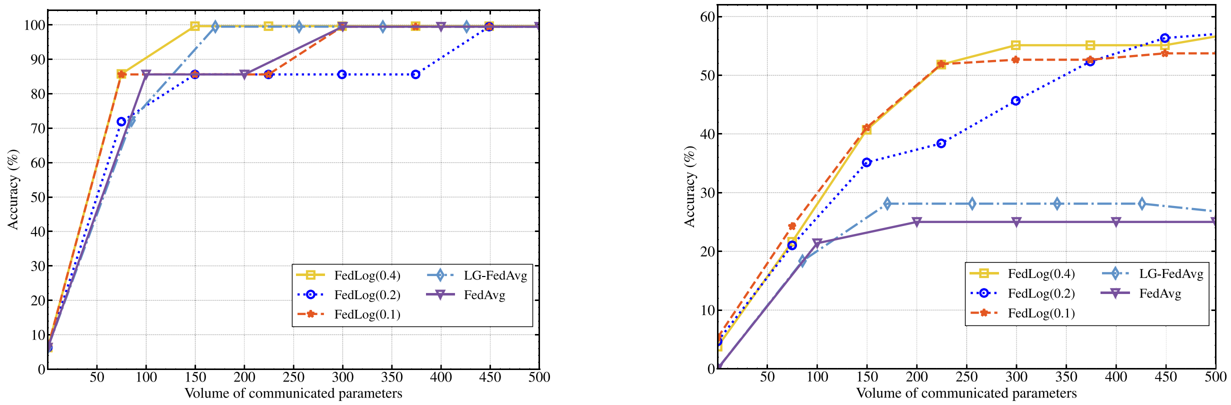
C. Performance Comparison With State-of-the-Art Federated Learning Frameworks

We then compare the performance of the proposed framework with baselines, i.e., Standalone, FedAvg [32], LG-FedAvg [25], under both IID and non-IID scenarios. Standalone refers to the scenario that each client builds the anomaly detection model alone, which is the ideal case for customization. We implement LG-FedAvg by keeping TCN and ACNN modules locally and sharing the MLP module. Further, we implement FedLog with varying pruning rate $r_p \in \{0.6, 0.7, 0.8, 0.9, 0.95, 0.99\}$ and name it with the proportion of unpruned parameters. For instance, FedLog with $r_p = 0.6$ (40% parameters unpruned) is denoted as FedLog(0.4).

To comprehensively evaluate the performance of different federated learning frameworks, we create a global test set with a balanced distribution of classes on the server side. We further divide each client's local dataset into the training, validation, and test set with a random 7:1:2 split. Note that the validation set is only used for local mask training. We conduct the following two tests on the local and global test sets:

- *Local Test:* This test is conducted to evaluate the customization of the proposed framework by testing each client's model on the local test set. We adopt the macro-average accuracy of all local models as the metric.
- *Global Test:* This test is conducted to evaluate the performance of the global model on the global test set. We adopt accuracy as the metric. Note that while evaluating Standalone and LG-FedAvg, we average all the client models as the global model, considering the fact that they do not produce a complete global model.

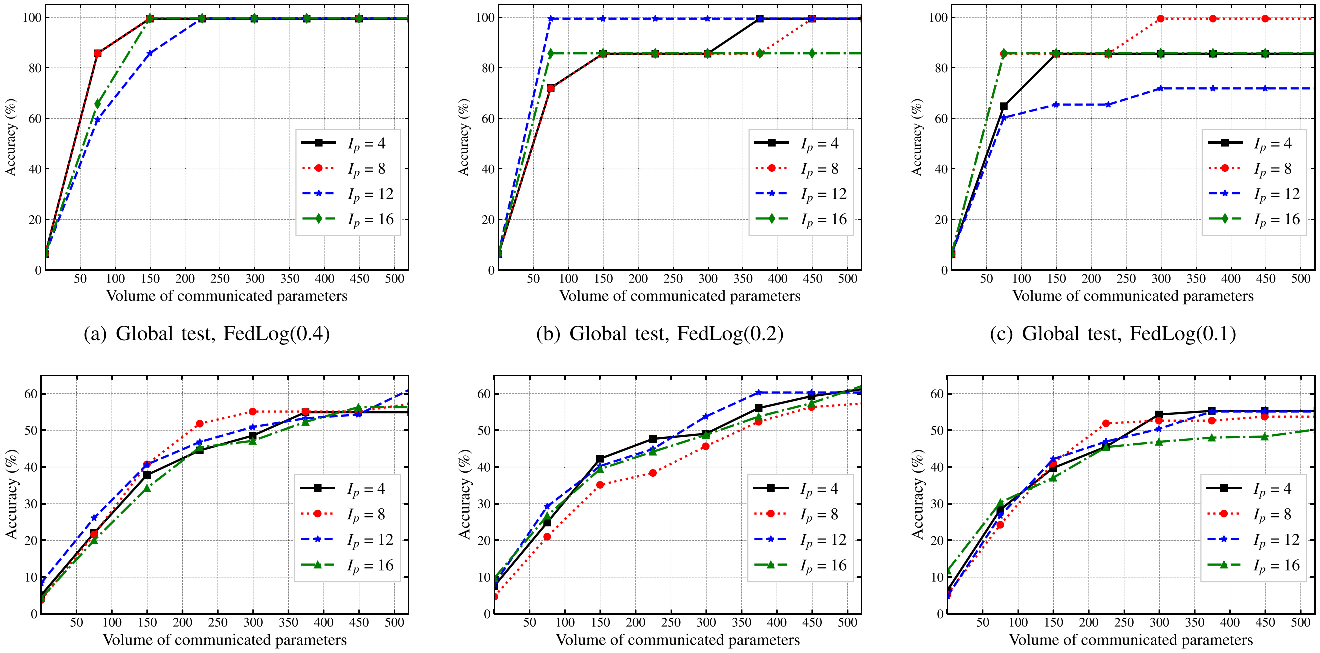
To demonstrate the superiority of FedLog, we first compare the best test results of the proposed FedLog with other frameworks within 50 communication rounds, which is shown in Table III. It can be easily seen that under the IID scenario, all the frameworks obtain a well-converged global model that shares similar performance. However, under the non-IID scenario, Standalone struggles in reaching a well-converged global model. Meanwhile, FedAvg and LG-FedAvg have a catastrophic performance, which is lower than 30%, in the local test. In contrast, the proposed FedLog not only shows the best performance in the global test, i.e., 99.63% of FedLog(0.4), but also produces comparable results in the local test, such as FedLog(0.1), FedLog(0.2), FedLog(0.3), and FedLog(0.4), compared to Standalone. As one can observe, FedLog exhibits the best performance under non-IID scenarios compared to other frameworks. This implies that FedLog provides the most customizable anomaly detection model. This is



(a) Global test

(b) Local test

Fig. 8. Comparison of accuracy with varying communication overheads in global and local tests.



(a) Global test, FedLog(0.4)

(b) Global test, FedLog(0.2)

(c) Global test, FedLog(0.1)

(d) Local test, FedLog(0.4)

(e) Local test, FedLog(0.2)

(f) Local test, FedLog(0.1)

Fig. 9. Performance comparison of different I_p with varying pruing rates in global and local tests.

because with the masking operation, LTN retains the essential weight parameters that could represent the specific distribution of the local dataset.

Second, we evaluate the communication efficiency of the proposed FedLog. Using Eq. (12), we can measure the communication cost of different federated learning frameworks. In FedAvg, $R = 1$, and in LG-FedAvg, $R = 0.85$. As for FedLog, the employed pruning rate $r_p \in \{0.6, 0.7, 0.8, 0.9, 0.95, 0.99\}$ corresponds to $R \in \{0.62, 0.56, 0.50, 0.43, 0.40, 0.38\}$, respectively. Table IV presents the numerical results about the communication overheads and accuracy of different frameworks under non-IID scenarios with $r = 5$, 10, 15, 20, 25. We can easily observe that FedLog(0.4) showcases the best global test accuracy in terms of all measured rounds. Additionally, though FedLog(0.05) and FedLog(0.01) converge slowly, all FedLogs reach a comparable local test accuracy with Standalone when $r = 25$. In comparison, FedAvg and LG-FedAvg suffer from catastrophic performance in the local test. Figure 8 intuitively displays the communication efficiency of different frameworks. It can be observed that the curve of FedLog(0.4) is above all other curves in Fig. 8(a), which means FedLog(0.4) is the most communication-efficient one in the global test. Similarly, as can be seen in Fig. 8(b), the proposed FedLog with any pruning rate considered outperforms LG-FedAvg and FedAvg in the local test in terms of communication efficiency. Thus, we can conclude that the proposed FedLog demonstrates the best communication efficiency over competitive federated learning frameworks.

Furthermore, we explore the impacts of pruning iterations I_p on FedLog's performance. Figure 9 visually presents the numerical results of the global and local test accuracy of all considered I_p with varying C under $r_p = 0.6, 0.8, 0.9$. It is clear that the choice of I_p could exert significant influence upon FedLog's performance in the global test. In contrast, I_p has limited impacts on FedLog's performance in the local test.

VII. Conclusion

In this paper, we proposed a customizable and communication-efficient federated anomaly detection scheme (FedLog) and demonstrated its capability to detect log anomalies in large-scale IoT systems. Specifically, our proposed scheme comprises a novel TCN-ACNN based deep anomaly detection model (to effectively extract fine-grained features from system logs and further facilitate anomaly identification), a new federated learning framework (to support IoT devices in establishing a comprehensive anomaly detection model in a collaborative and privacy-preserving manner), and a lottery ticket hypothesis-based masking strategy (to significantly reduce communication overheads and achieve customizable federated learning in handling non-IID log datasets).

While the performance evaluation findings using both HDFS [1] and BGL [37] datasets suggested that FedLog outperforms two other competing log-based anomaly detection models [7], [8] under both centralized learning and federated learning paradigms, there are a number of potential extensions to this work. For example, the cyberthreat landscape in practice may differ significantly from those captured in the datasets (even though both HDFS and BGL datasets are widely used in the research community). Hence, we intend to implement a prototype of the proposed FedLog in a real-world setting, for example across several collaborating universities, to evaluate its real-world utility and performance.

References

[1] W. Xu, L. Huang, A. Fox, D. Patterson, and M. I. Jordan, "Detecting large-scale system problems by mining console logs," in *Proc. ACM Symp. Oper. Syst. Principles (SOSP)*, Oct. 2009, pp. 117–132.

[2] P. He, J. Zhu, S. He, J. Li, and M. R. Lyu, "Towards automated log parsing for large-scale log data analysis," *IEEE Trans. Dependable Secure Comput.*, vol. 15, no. 6, pp. 931–944, Nov./Dec. 2018.

[3] S. Han *et al.*, "Log-based anomaly detection with robust feature extraction and online learning," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 2300–2311, 2021.

[4] M. A. M. Ahsan, A. W. B. A. Wahab, M. Y. I. B. Idris, S. Khan, E. Bachura, and K. R. Choo, "CLASS: Cloud log assuring soundness and secrecy scheme for cloud forensics," *IEEE Trans. Sustain. Comput.*, vol. 6, no. 2, pp. 184–196, Apr.–Jun. 2021.

[5] Y. Luo, Y. Xiao, L. Cheng, G. Peng, and D. D. Yao, "Deep learning-based anomaly detection in cyber-physical systems: Progress and opportunities," *ACM Comput. Surveys*, vol. 54, no. 5, p. 106, 2022.

[6] G. Pang, C. Shen, L. Cao, and A. Van Den Hengel, "Deep learning for anomaly detection: A review," *ACM Comput. Surveys*, vol. 54, no. 2, p. 38, 2022.

[7] M. Du, F. Li, G. Zheng, and V. Srikumar, "DeepLog: Anomaly detection and diagnosis from system logs through deep learning," in *Proc. ACM Conf. Comput. Commun. Security (CCS)*, Oct./Nov. 2017, pp. 1285–1298.

[8] W. Meng *et al.*, "LogAnomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs," in *Proc. Int. Joint Conf. Artif. Intell. (IJCAI)*, vol. 7, Aug. 2019, pp. 4739–4745.

[9] X. Zhang *et al.*, "Robust log-based anomaly detection on unstable log data," in *Proc. ACM Joint Meeting Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng. (ESEC/FSE)*, Aug. 2019, pp. 807–817.

[10] S. Huang, Y. Liu, C. Fung, R. He, Y. Zhao, H. Yang, and Z. Luan, "HitAnomaly: Hierarchical transformers for anomaly detection in system log," *IEEE Trans. Netw. Service Manag.*, vol. 17, no. 4, pp. 2064–2076, Dec. 2020.

[11] C. Qiu, T. Pfrommer, M. Kloft, S. Mandt, and M. Rudolph, "Neural transformation learning for deep anomaly detection beyond images," in *Proc. Int. Conf. Mach. Learn. (ICML)*, vol. 139, Jul. 2021, pp. 8703–8714.

[12] L. Deecke, L. Ruff, R. A. Vandermeulen, and H. Bilen, "Transfer-based semantic anomaly detection," in *Proc. Int. Conf. Mach. Learn. (ICML)*, vol. 139, Jul. 2021, pp. 2546–2558.

[13] H. N. Noura, O. Salman, A. Chehab, and R. Couturier, "DistLog: A distributed logging scheme for IoT forensics," *Ad Hoc Netw.*, vol. 98, Mar. 2020, Art. no. 102061.

[14] 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 Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2019, pp. 756–767.

[15] Y. Liu *et al.*, "Deep anomaly detection for time-series data in industrial IoT: A communication-efficient on-device federated learning approach," *IEEE Internet Things J.*, vol. 8, no. 8, pp. 6348–6358, Apr. 2021.

[16] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-i.i.d. data," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 9, pp. 3400–3413, Sep. 2020.

[17] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of FedAvg on non-IID data," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, Apr. 2020, pp. 1–26.

[18] A. Li *et al.*, "LotteryFL: Personalized and communication-efficient federated learning with lottery ticket hypothesis on non-IID datasets," Aug. 2020, *arXiv:2008.03371*.

[19] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, May 2019, pp. 1–42.

[20] P. He, J. Zhu, S. He, J. Li, and M. R. Lyu, "An evaluation study on log parsing and its use in log mining," in *Proc. Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, vol. 1, Jun. 2016, pp. 654–661.

[21] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," 2016, *arXiv:1610.05492*.

[22] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-IID data," Jun. 2018, *arXiv:1806.00582*.

[23] T. Shen *et al.*, "Federated mutual learning," Jun. 2020, *arXiv:2006.16765*.

[24] Y. Zhang, T. Xiang, T. M. Hospedales, and H. Lu, "Deep mutual learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2018, pp. 4320–4328.

[25] P. P. Liang *et al.*, "Think locally, act globally: Federated learning with local and global representations," Jul. 2020, *arXiv:2001.01523*.

[26] E. Diao, J. Ding, and V. Tarokh, "HeteroFL: Computation and communication efficient federated learning for heterogeneous clients," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, May 2021, pp. 1–24.

[27] S. He, J. Zhu, P. He, and M. R. Lyu, "Experience report: System log analysis for anomaly detection," in *Proc. IEEE Int. Symp. Softw. Reliab. Eng. (ISSRE)*, Oct. 2016, pp. 207–218.

[28] A. Dridi, C. Boucetta, S. E. Hammami, H. Afifi, and H. Moungla, "STAD: Spatio-temporal anomaly detection mechanism for mobile network management," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 1, pp. 894–906, Mar. 2021.

[29] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surveys*, vol. 41, no. 3, pp. 1–58, Jul. 2009.

[30] B. Li, Y. Wu, J. Song, R. Lu, T. Li, and L. Zhao, "DeepFed: Federated deep learning for intrusion detection in industrial cyber–physical systems," *IEEE Trans. Ind. Informat.*, vol. 17, no. 8, pp. 5615–5624, Aug. 2021.

[31] B. Li, P. Wang, H. Huang, S. Ma, and Y. Jiang, "FlPhish: Reputation-based phishing byzantine defense in ensemble federated learning," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Sep. 2021, pp. 1–6.

[32] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Int. Conf. Artif. Intell. Stat. (AISTATS)*, Apr. 2017, pp. 1273–1282.

[33] J. Kiefer and J. Wolfowitz, "Stochastic estimation of the maximum of a regression function," *Ann. Math. Stat.*, vol. 23, pp. 462–466, Sep. 1952.

[34] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," Mar. 2018, *arXiv:1803.01271*.

[35] Y. LeCun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *Proc. Int. Conf. Neural Inf. Process. Syst. (NeurIPS)*, Nov. 1990, pp. 598–605.

[36] H. Song, P. Jeff, T. John, and J. D. William, "Learning both weights and connections for efficient neural network," in *Proc. Int. Conf. Neural Inf. Process. Syst. (NIPS)*, vol. 1, Dec. 2015, pp. 1135–1143.

[37] A. Oliner and J. Stearley, "What supercomputers say: A study of five system logs," in *Proc. Int. Conf. Dependable Syst. Netw. (DSN)*, Jun. 2007, pp. 575–584.

[38] P. Kairouz *et al.*, "Advances and open problems in federated learning," Dec. 2019, *arXiv:1912.04977*.

Beibei Li (Member, IEEE) received the B.E. degree in communication engineering from the Beijing University of Posts and Telecommunications, China, in 2014, and the Ph.D. degree in cybersecurity from Nanyang Technological University, Singapore, in 2019.

He is currently an Associate Professor (Doctoral Supervisor) with the School of Cyber Science and Engineering, Sichuan University, China. He was invited as a Visiting Researcher with the Faculty of Computer Science, University of New Brunswick, Canada, from March to August 2018. His research interests span several areas in security and privacy issues on cyber–physical systems (e.g., smart grids, industrial control systems, and intelligent transportation systems), with a focus on intrusion detection techniques, artificial intelligence, and applied cryptography. He is serving or has served as the Publicity Chair, the Publication Co-Chair, the Track Chair, or a TPC member for several international conferences, including IEEE ICC, IEEE GLOBECOM, AAAI, IEEE ICNC, and PST. His works have been published in IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, IEEE TRANSACTIONS ON POWER SYSTEMS, *ACM Transactions on Cyber-Physical Systems*, IEEE INTERNET OF THINGS JOURNAL, *Automatica*, *Information Sciences*, IEEE ICC 2021, IEEE GLOBECOM 2021, ICSOC 2021, and IEEE ISCC 2021 (with the Best Paper Award).

Shang Ma (Student Member, IEEE) is currently pursuing the B.E. degree in cybersecurity with the School of Cyber Science and Engineering, Sichuan University, Chengdu, China. His research interests are anomaly detection, natural language processing, and federated learning.

Ruilong Deng (Senior Member, IEEE) received the B.Sc. and Ph.D. degrees in control science and engineering from Zhejiang University, Hangzhou, Zhejiang, China, in 2009 and 2014, respectively.

He was a Research Fellow with Nanyang Technological University, Singapore, from 2014 to 2015; an AITF Postdoctoral Fellow with the University of Alberta, Edmonton, AB, Canada, from 2015 to 2018; and an Assistant Professor with Nanyang Technological University from 2018 to 2019. He is currently a Professor with the College of Control Science and Engineering, Zhejiang University, where he is also affiliated with the School of Cyber Science and Technology. His research interests include cyber security, smart grid, and communication networks.

Dr. Deng serves/served as an Associate Editor for IEEE TRANSACTIONS ON SMART GRID, IEEE POWER ENGINEERING LETTERS, IEEE/CAA JOURNAL OF AUTOMATICA SINICA, and IEEE/KICS JOURNAL OF COMMUNICATIONS AND NETWORKS and a Guest Editor for IEEE TRANSACTIONS ON EMERGING TOPICS IN COMPUTING, IEEE TRANSACTIONS ON CLOUD COMPUTING, and *IET Cyber–Physical Systems: Theory & Applications*. He also serves/served as the Symposium Chair for IEEE SmartGridComm'19 and IEEE GLOBECOM'21.

Kim-Kwang Raymond Choo (Senior Member, IEEE) currently holds the Cloud Technology Endowed Professorship with the University of Texas at San Antonio, San Antonio, TX, USA. He has received the Outstanding Editor Award for 2021 from Future Generation Computer Systems. He has also received Best Paper Awards from the IEEE Systems Journal in 2021, the IEEE Computer Society's Bio-Inspired Computing STC Outstanding Paper Award for 2021, the IEEE DSC 2021, the *IEEE Consumer Electronics Magazine* for 2020, the *Journal of Network and Computer Applications* for 2020, *EURASIP Journal on Wireless Communications and Networking* in 2019, IEEE TrustCom 2018, and ESORICS 2015; the IEEE Blockchain 2019 Outstanding Paper Award; and the Best Student Paper Awards from Inscrypt 2019 and ACISP 2005. He was a recipient of the 2019 IEEE Technical Committee on Scalable Computing Award for Excellence in Scalable Computing (Middle Career Researcher), the British Computer Society's 2019 Wilkes Award Runner-up, the Fulbright Scholarship in 2009, the 2008 Australia Day Achievement Medallion, and the British Computer Society's Wilkes Award in 2008. He is the Founding Co-Editor-in-Chief of ACM Distributed Ledger Technologies: Research and Practice and the Founding Chair of IEEE TEMS Technical Committee on Blockchain and Distributed Ledger Technologies. He is an ACM Distinguished Speaker and IEEE Computer Society Distinguished Visitor from 2021 to 2023 and a Web of Science's Highly Cited Researcher (Computer Science 2021, Cross-Field 2020).

Jin Yang received the M.S. and Ph.D. degrees in computer science from Sichuan University, Sichuan, China, in 2004 and 2007, respectively, where he is currently an Associate Professor with the School of Cyber Science and Engineering. His main research interests include network security, knowledge discovery, and expert systems.