Hindawi
Mathematical Problems in Engineering
Volume 2022, Article ID 9031701, 10 pages
https://doi.org/10.1155/2022/9031701

Research Article

A Federated Learning-Based Fault Detection Algorithm for Power Terminals

Shuai Hou, Jizhe Lu, Enguo Zhu, Hailong Zhang, and Aliaosha Ye

China Electric Power Research Institute, Haidian District, Beijing 100192, China

Correspondence should be addressed to Shuai Hou; houshuai@epri.sgcc.com.cn

Received 7 May 2022; Accepted 21 June 2022; Published 19 July 2022

Academic Editor: Zhihan Lv

Power terminal is an important part of the power grid, and fault detection of power terminals is essential for the safety of the power grid. Existing fault detection of power terminals is usually based on artificial intelligent or deep learning models in the cloud or edge servers to achieve high accuracy and low latency. However, these methods cannot protect the privacy of the terminals and update the detection model incrementally. A terminal-edge-server collaborative fault detection model based on federated learning is proposed in this study to improve the accuracy of fault detection, reduce the data transmission and protect the privacy of the terminals. The fault detection model is initially trained in the server using historical data and updated using the parameters of local models from edge servers according to different updating strategies, then the parameters will be sent to each edge server and further to all terminals. Each edge server updates the local model via the compressed system log from terminals in its coverage region, and each terminal uses the model to detect fault according to the system behavior in the log. Experiment results show that this fault detection algorithm has high accuracy and low latency, and the accuracy increases with more model updating.

1. Introduction

With the development of the power grid and the applications of information and communication technologies, the smart grid has been widely deployed in most countries, and an automated and distributed advanced energy delivery network has also been constructed. Nowadays, the power terminals, such as smart energy meter, concentrator, special transformer, and energy controller, have become more and more intelligent and important in the smart grid, and the reliability, security, and stability of the power terminals have been challenges for the smart grid. Once the power terminal fails, it will lead to inaccurate power data, confused power scheduling, and damage of power equipment and even part of the power grid. Hence, the fault detection of power terminals with high accuracy is needed to find the faults quickly to avoid the damage to the power grid.

Fault detection is the foundation of the security of power grid. At present, there are many fault detection methods for power terminals, most of which use artificial intelligent or deep learning models to improve the accuracy of fault detection. In these methods, the detection operation is executed on the servers and the state data generated on the terminal of collected by extra devices are transmitted to the server. For example, drone and auto-tracking camera were used to detect the defects in power lines in reference [1]. Bouazza et al. [2] proposed artificial intelligence-based methods to detect faults of the power switches in the wind energy conversion system. Shi et al. [3] proposed a fault detection method based on the LSTM model to predict the faults of DC-DC power supply. Nguyen [4] used a microphone to detect the overload of large power transformer by sound analysis. Hong et al. [5] detected the open conductor fault in power distribution networks using multiple measurement factors of feeder RTUs with DGs. Visible images, infrared images, and ultraviolet images of power equipment were fused to train a deep learning-based fault detection model [6] in a power system. Improved random forest was used to detect the power outage accident of the power terminal in reference [7], and continuous wavelet transform and convolution neural network were adopted to detect faults for power electronic converters in reference [8].

The above fault detection methods can achieve high accuracy mainly by the complex model and massive data. But the complex model will lead to high requirements of computing and storage resource, and massive data may result in congestion and delay of network transmission. Therefore, the fault detection should be at the terminal and the model training should be in the cloud to make full use of the advantages of cloud computing resources and local data to take the accuracy, efficiency, and privacy of fault detection into consideration.

Edge computing is a new framework to provide service at the edge of the network [9, 10]. It has been used in many applications, especially for fault detection of IoT (Internet of Things) devices. An IoT fault detection based on edge computing and blockchain was proposed in reference [11], in which weighted random forest was adopted. Huong et al. developed LocKedge [12] framework to detect low complexity cyberattack in IoT edge computing. Mishra et al. [13] proposed data anomalies detection method at the edge of pervasive IoT systems. A device-edge split architecture for intrusion detection for IoT devices was proposed in reference [14] to reduce the overhead of the IoT devices. The architecture of the power grid is similar to the framework of edge computing, hence researchers have attempted to apply edge computing in the power grid for fault detection. Huo et al. [15] proposed a fault detection based on edge computing for distributed power distribution, and a method based on edge computing architecture was proposed to judge unsafe actions of electric power operations in time in reference [16]. Yang et al. [17] proposed a semisupervised cloud edge collaborative unsafe actions detection framework, and Zhang et al. [18] combined cloud edge fusion framework and deep learning techniques for abnormal object detection in the power grid. These studies overcome the accuracy requirement of fault detection in the power grid by splitting the fault detection to edge server or cloud to reduce the overhead of the power terminals; however, the data privacy of the terminals and the data transmission in the network are not well considered.

Federated learning [19], which is first proposed in 2016, is a machine learning method that takes into account model accuracy and data privacy. It has been used in the power grid to protect the privacy of the data of consumers or terminals. A privacy-preserving federated learning framework Fed-Detect [20] is developed for energy theft detection in smart grid, and a federated learning-based method was proposed for privacy-preserving household characteristic identification in reference [21]. Su et al. [22] proposed a secure and efficient federated-learning-enabled AIoT scheme for private energy data sharing in smart grids with edge-cloud collaboration. Wang et al. [23] developed a distributed electricity consumer characteristics identification method based on federated learning to preserve the privacy of retailers. Liu et al. [24] used an asynchronous decentralized federated learning model for collaborative fault diagnosis of PV stations. These methods protect the privacy of the power terminals but cannot be applied to fault detection because of requirements of the high accuracy, low latency, data privacy, and incremental updating.

To overcome the above problems, we propose a three-tier fault detection model based on federal learning for power terminals and three different model updating strategies in this study. The fault detection model training, updating, and testing are split into edge servers, cloud, and terminals respectively in the power grid. The cloud is responsible for the construction of the initial fault detection model and subsequent model updating to improve the accuracy of fault detection; the edge server is responsible for training the local model to ensure the data privacy of the terminal, and the terminal that only uses the trained model for fault detection to improve the efficiency of fault detection can compress the raw system log to reduce the data transmission. In the data interaction between terminal, edge server, and cloud, the amount of data transmission and system delay is reduced by means of log compression and parameter transmission. The experimental results show that our algorithm can reduce the amount of data transmission and achieve higher accuracy than traditional fault detection methods.

2. Edge-Cloud Collaboration Fault Detection Framework

2.1. Data Model of Power Terminal. The functions and manufacturers of power terminals are different in the power grid, and the configurations of hardware resources are also different. The configuration of some typical power terminals is listed in Table 1. The complex fault detection model cannot be trained on power terminals, but fault detection using the trained model can be performed on most terminals.

The faults of the power terminals involve various functional modules of the embedded operating system, such as hardware driver, system security, file system, system application, and memory management. When a system fault occurs, the system log will record the fault-related information, including the fault occurrence time and system abnormal behaviors. Therefore, the fault detection can be realized through the analysis of the operating system log. Though log formats and the description of system behaviors of different embedded operations are different, the system log usually includes system time, system components, and behavior description. Figure 1 shows the log format of an embedded operating system designed independently, and the record of the system log is in the form of time, device, and detail, where time and device represent the system time and the system component of this system behavior respectively, and detail is the operation description of the system behavior. As shown in the first line in Figure 1, "3.567604" is the relative time after system startup, "USB usb3" is the system component, and "Manufacture: Linux 3.10.108 ohci_hcd" is the system description.

The system fault detection of power terminal needs to find the fault and judge the type of fault as soon as possible after it happens, so that the users can quickly deal with the fault to avoid serious accidents, such as terminal hardware damage, data loss, and network intrusion. When the system fails, the *device* and *detail* attributes of the system log record

Table 1: Configurations of typical power terminals.

	Smart meter	Special transformer	Fusion terminal	Energy controller
Frequency	≥120 MHz	≥300 MHz	≥1 GHz	≥1.2 GHz
RAM	≥512 KB	≥512 MB	≥2G B	≥2 GB
Flash (code)	≥512 KB	≥128 MB	≥8 GB	≥4 GB
Flash (storage)	≥8 MB			
Memory protection	MPU (memory protection unit)	MMU (memory management unit)	MMU (memory management unit)	MMU (memory management unit)
Architecture	Arm Cortex-M4	ARM 920T™	ARM Cortex™-A7	ARM Cortex™-A7

```
[3.567604] usb usb3: Manufacturer: Linux 3.10.108 ohci_hcd
[3.573805] usb usb3: SerialNumber: sunxi-ohci
[3.579445] hub 3-0:1.0: USB hub found
[3.583635] hub 3-0:1.0: 1 port detected
[3.588858] usbcore: registered new interface driver usb-storage
[3.595695] usbcore: registered new interface driver ums-alauda
[3.602414] usbcore: registered new interface driver ums-cypress
[3.609211] usbcore: registered new interface driver ums-datafab
```

Figure 1: Example of system log of power terminals.

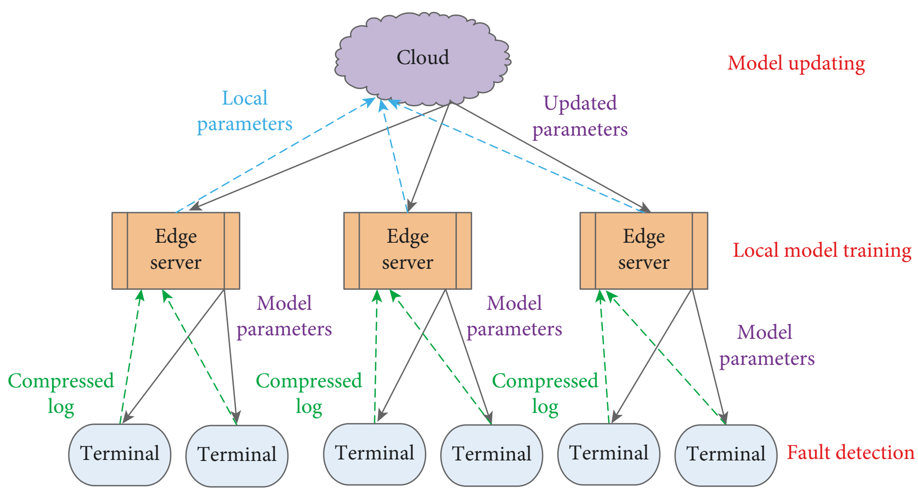

Figure 2: Three-tier fault detection architecture.

the abnormal behavior or operations of the system devices. Some common system faults can be found through the keywords in the *detail* attribute of the system log, and the system log usually contains some fault-sensitive words such as failed, error, and warning, as listed in Table 2. However, some system faults caused by hidden bugs or premeditated external attacks may not contain these fault-sensitive words, and the specific type of the system fault is difficult to determine by keywords. Therefore, it is necessary to analyze the system log using natural language processing methods to accurately identify and classify the faults.

Though the system log records the system behaviors and operations when the system failure occurs, the content of the system log is generally simplified in the embedded operating system. Therefore, it is necessary to take the language characteristics of the system log into consideration when using natural language processing methods:

(1) *Short Sentence*: Each record of the system log has less content, the fault location is usually a specific word or number of the system component, and the fault description is usually a short sentence, which contains only a few words. The syntax structure of the sentence is simple.

(2) *Less Vocabulary*. There are less vocabularies in the whole system log. The frequencies of most words in the system log are high, while some words with lower frequency are usually related to the identification of the terminal and have nothing to do with the type of the fault, such as specific IP addresses.

Table 2: Examples of sensitive words.

ID	Detail of the system log
1	Warning: get ephy clock is failed
2	Warning: mountpoint for pids not found
3	Console-setup.service: failed with result "exit-code"
4	Failed to start: set console font and keymap
5	Sunxi-ahci: Probe of sata failed with error −1
6	Error while tracing: no such file or directory

(3) *High Redundancy*. The system log generated by each terminal contains a large number of repeated records, and the system logs generated by similar terminals also have a large amount of redundancy. The redundant records are typically representing normal operations.

Therefore, in order to improve the efficiency of fault detection by natural language processing, the redundant records in the system log should be removed to reduce the number of records to be processed. In the process of recognition, the characteristics of short sentences and less vocabulary should be taken into consideration to reduce the complexity of the detection model.

2.2. Fault Detection Framework. Considering the data model and hardware configuration of power terminals, the requirements of fault detection accuracy and delay, as well as the network architecture of power grid, a three-tier fault detection architecture, namely, end-edge-cloud, for power terminals is proposed in this study, as shown in Figure 2. The fault detection model is trained initially and updated in the cloud, and the terminals detect the fault using the detection model. Each edge server is used to train the local model using data from the terminals in its coverage region to protect the privacy of the terminals and generate the parameters of the updated model for the cloud.

(1) *Power Terminals*. They are deployed to perform measuring, monitoring, controlling, and other functions in the power grid, such as energy controllers, fusion terminals, intelligent meters, and special transformer terminals. The power terminals are designed, manufactured, and used by different companies or end users, and they typically have low hardware configuration, and perform different tasks by running different application software. The faults of the power terminal and the abnormal behaviors resulted by the faults are recorded in the system log, hence the fault detection should be performed on the power terminal to detect the fault as soon as possible. Power terminals are the main participants of fault detection and are responsible for the collection and preprocessing detect system log and the detection of the faults.

(2) *Edge Servers*. They can be either the dedicated servers or powerful terminals, which have a large number of computing and storage resources and can perform the collection and processing of large datasets. Each edge server is responsible for the collection of system logs and the training of local fault detection model for multiple terminals in a distinct region. The edge server acts as the role of connector of the terminals and the cloud in the fault detection architecture, and they interact with the cloud for the parameters of fault detection model, terminals for the updated parameters of the fault model, and the compressed system logs.

(3) *Cloud*. With powerful computing power and storage resources, it is deployed and maintained by the power grid company and is responsible for the initialization and update of the global fault detection model. In practice, one or several large data centers are constructed as the cloud of the power grid, and they are the control and decision centers of the power grid.

The end-edge-cloud three-tier architecture is the common architecture of the practical power grid, hence the three-tier fault detection model can be easily deployed and completed without extra hardware support. Due to the limited network transmission capacity of the terminals, the bandwidth constraints of the edge server, the long transmission distance between the edge server and the cloud, and the data size and delay of data transmission between the cloud, edge server, and terminals greatly affect the training and detection performance of the fault detection model. Therefore, the amount of data transmission between terminals, edge server, and the cloud should be minimized to improve fault detection performance.

From the perspective of model training, the cloud performs the training of the initial fault detection model and the updating of model. The edge server is responsible for collecting the compressed system log of the terminals and training the local fault detection model. The terminal uses the trained model to detect the system fault. The powerful computing resource of the cloud can quickly complete the model training and updating. The local model training can also be quickly accomplished with the small dataset on the edge server. The fault detection on the terminals only has a little demand for computing resources.

From the perspective of data transmission, the system log generated by the terminal will be compressed and transmitted to the adjacent edge server, which greatly reduces the amount of data and occupies less network bandwidth. The edge server transmits the parameters of the local model to the cloud and the parameters of updated model to terminals. Although the distance between the edge server and cloud is long, the transmission delay is very small because the parameters of the model is far less than the raw system logs.

3. Federated Learning-Based Fault Detection Algorithm

The spirit of federated learning is used in the three-tier fault detection framework to reduce the data transmission between the cloud and edge servers and protect the privacy of power terminals. The fault detection algorithm consists of three steps: pretraining, local training, and model updating. The pretraining step is processed in the cloud, the local training is completed on the edge server, and the model updating is finished in the cloud.

3.1. Pretraining. The cloud uses historical system logs to train the initial fault detection model, and the system logs are marked and provided by the manufacturers and managers. Each record in the system log is in the form of device, time, detail, and result, where device is the type of the terminal; time, device, and detail are the system time, system component, and system behavior of the fault respectively; *result* is a nonnegative integer and represents the type of the fault. The natural language processing method is adopted to recognize the fault from the system behavior in the system log. The description of the system behavior is recorded in the system log in the form of detail = $(w1, w2, w3, \ldots, wm)$, where m is the number of words and w_i is the ith word, $1 \leq i \leq m$.

The LSTM (long short-term memory) network model is used in our fault detection model to improve the accuracy of fault detection. LSTM is an improvement of RNN (recurrent neural network), and it uses an input gate, forget gate, and output gate to selectively retain part of the previous cell state and transfer it to the next cell to overcome the problem of long-term dependence in RNN. Each cell state in LSTM is consisted of one forget gate, one input gate, and one output gate, and each gate is composed of a sigmoid neural network layer and a pointwise operation, as shown in Figure 3.

The forget gate is used to forget part of the information of the previous cell state C_{t-1}, and only $f_t * C_{t-1}$ is remained in the current cell state, where f_t is the output coefficient of a sigmoid layer:

$$f_t = \text{sigmoid}\left(w_f\left[h_{t-1}, x_t\right] + b_f\right). \tag{1}$$

The input gate is used to determine which new information will be keep in the current cell state, and the new information is defined as $i_t * \widetilde{C}_t$, where i_t is the output of sigmoid layer and $\widetilde{C}_t$ is the output of a tanh layer:

$$\begin{aligned} i_t &= \text{sigmoid}\left(w_i\left[h_{t-1}, x_t\right] + b_i\right), \\ \widetilde{C}_t &= \tanh\left(w_c\left[h_{t-1}, x_t\right] + b_c\right). \end{aligned} \tag{2}$$

The current cell state C_t can be obtained after the forget gate and input gate:

$$C_t = f_t * C_{t-1} + i_t * \widetilde{C}_t. \tag{3}$$

The output gate is used to generate the output h_t of current cell state C_t. The output coefficient o_t and tanh will be computed to get the output h_t:

$$\begin{aligned} o_t &= \text{sigmoid}\left(w_0\left[h_{t-1}, x_t\right] + b_0\right), \\ h_t &= o_t \tanh\left(C_t\right). \end{aligned} \tag{4}$$

3.2. Local Training. After completing the LSTM model training, the cloud sends the parameters of the fault detection model to each edge server, which then sends the parameters to all terminals in its corresponding region. During the execution of the terminal, the system log will be preprocessed and input into the fault detection model to obtain the fault detection results, and finally, the terminal carries out corresponding response, such as restart, shutdown, and alarm. Because only fault detection is performed on the terminal, and the model training and updating are performed on the edge server and cloud, the terminal can quickly detect and respond the fault.

Due to the high reliability and security requirements of power grid, the failure probability of each terminal is very low, most records in the system log are normal operations, and only a few records are abnormal system behavior in a long time. Hence, the system log on each terminal contains a lot of redundant information, and it should be compressed before being transmitted to the edge server to reduce the amount of data transmission. Log compression is mainly divided into three steps:

① *Variable Replacement*. Some variables in the system log represent the identification of the device or network and do not contain the semantics of the fault. These variables are usually long words or strings and can be replaced with the category of these variables, such as IP address and web page address, and device name can be replaced with IP, URL, and local, so as to reduce the differences between records in the system log and reduce the data transmission between the terminal and edge server.

② *Similarity Computing*. Each record of the system log contains the system component and the system behavior of the fault, and the records with high similarity will be compressed into one record to reduce the data transmission. Hence, the similarity of two records consists of the similarity of the system component S_{device} and that of the system behavior S_{detail}. The similarity of two system components depends on the locations of the components in the fault tree:

$$\begin{gathered} S_{\text{device}}\left(\text{device}_1, \text{device}_2\right) \\ = \begin{cases} 1, & \text{device}_1 = \text{device}_2 \\ \dfrac{1}{\text{max length}} & \text{device}_1 \neq \text{device}_2 \end{cases}, \\ \text{maxlength} = \max\left\{P\left(\text{Root}, \text{device}_1\right), P\left(\text{Root}, \text{device}_2\right)\right\}, \end{gathered} \tag{5}$$

where Root is the nearest common ancestor node of device_1 and device_2 in the fault tree; $p(\text{Root}, \text{device}_1)$ and $p(\text{Root}, \text{device}_2)$ are the number of nodes on the path from Root to device_1 and device_2, respectively. If the detail of the record in the system log contains one or more of words in the set of sensitive words Keywords, then the record is marked as a sensitive record, and the similarity of this record will not be computed because any sensitive record cannot be compressed to ensure the accuracy of the fault

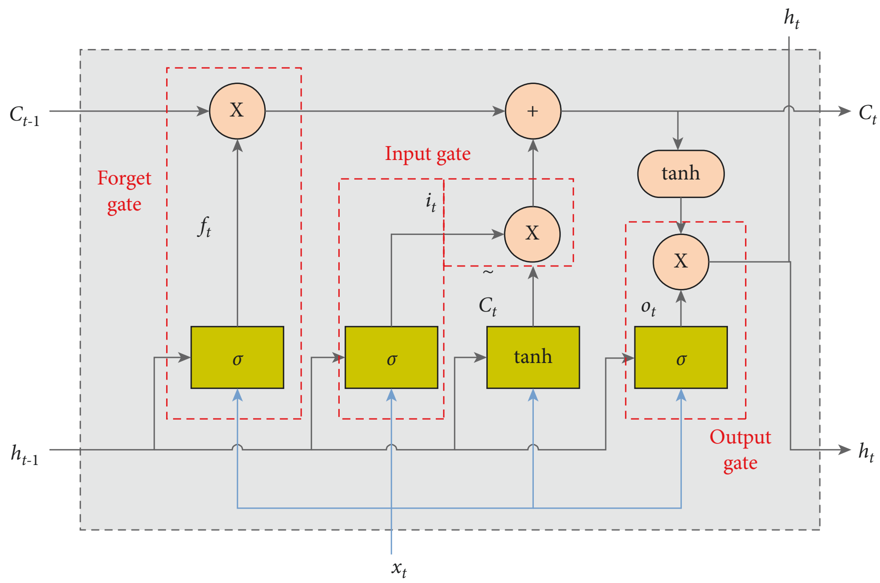

Figure 3: Structure of cell state in LSTM.

detection. For the nonsensitive records, the similarity of the system behavior is related to the words in detail. Given $\text{details}_1\,(w_1, w_2, \ldots, w_m)$ and $\text{details}_2\,(w_1', w_2', \ldots, w_n')$, the similarity of details_1 and device_2 can be obtained:

$$S_{\text{detail}}\left(\text{detail}_1, \text{detail}_2\right) = \frac{\left|\text{detail}_1 \cap \text{detail}_2\right|}{\left|\text{detail}_1 \cup \text{detail}_2\right|}, \tag{6}$$

where m and n are the number of words in detail_1 and detail_2 respectively, $\text{details}_1 \cap \text{details}_2$ is the number of words in the intersection of detail_1 and detail_2, and $\text{details}_1 \cup \text{details}_2$ are the number of words in the union of detail_1 and detail_2.

The similarity of two records $r_1 = (\text{time}_1, \text{device}_1, \text{details}_1)$ and $r_2 = (\text{time}_2, \text{device}_2, \text{details}_2)$ in the system log $S(r_1, r_2)$ is the weighted sum of the similarity of S_{device} and S_{details}:

$$\begin{aligned} S\left(r_1, r_2\right) = {} & \alpha \cdot S_{\text{device}}\left(\text{device}_1, \text{device}_2\right) \\ & + \beta \cdot S_{\text{detail}}\left(\text{detail}_1, \text{detail}_2\right), \end{aligned} \tag{7}$$

where α and β are the coefficients of S_{device} and S_{details} respectively and can be set by the domain experts, such as system designers, maintainers, and testers.

③ *Redundancy Filtering*. Log records with high similarity generated in a short time are called redundant records. The redundant logs should be discarded on the terminal, and only the remaining log records will be transmitted to the edge server to reduce the data transmission. The filtering process is as follows: first, all the sensitive log records should be retained, that is, the record $r_i = (\text{time}_i, \text{device}_i, \text{details}_i)$ with $\text{details}_i = (w_1, w_2, \ldots, w_m)$ will be transmitted to the edge server if $w_j \in \text{detail}_j$ and $w_j \in \text{Keywords}_j$. If r_j is not a sensitive record, then its similarity with any nonsensitive record r_j, $S(r_i, r_j)$, is calculated in the subsequent short time interval. The log record r_j with $S(r_i, r_j) \geq S_{th}$ will be marked as redundant record and discarded on the terminal, where S_{th} is the similarity threshold set by the system manager or other experts.

After receiving the compressed system logs from terminals in its region, the edge server performs the local training of the fault detection model using the data received. Then, the parameters of the updated local model are transmitted to the cloud to update the global fault detection model. Due to the low probability of system failure of each terminal, the local training should not be performed frequently. For example, a manufacturer of power terminal found that the system failure of each terminal occurs once every two months during on-site use, according to a pilot testing of energy controller since April 2021. If the fault detection model is trained frequently using the dataset without fault records, then it may lead to overfitting of the fault detection model. In order to ensure the detection accuracy of fault detection model, three strategies of local training are proposed in this study.

① *Periodic Update (PUpdate)*. Each edge server trains its local model within a specified period. The parameter period is set according to the reliability of the terminals in the coverage region of the edge server, and it is set to be the average failure time of the terminals by default. Given the edge server server_i and the set of terminals in its coverage region $P_i = (p_{i,1}, p_{1,2}, \ldots p_{i,k})$, the failure time of the terminal $p_{i,j}$ is ft_j, and we can obtain the update period of edge server server_i as follows:

$$\text{period}_i = \frac{1}{k} \sum_{j=1}^{k} ft_i, \tag{8}$$

where k is the number of terminals. The update period of different edge servers may also be different and can be adjusted when the terminals in the region are removed or newly deployed.

② *Incremental Update (AUpdate).* Each edge server starts the local training when the number of abnormal log records is no less than the threshold. Since training the local model using normal records may lead to over fitting, the training dataset should contain some abnormal records. Suppose is the training dataset on edge server server_i, D^{th} is the threshold of the abnormal records, the server_i will perform local training if $|\text{ADataset}_i| \geq Dt\ h$, where NDataseti is the set of normal records in Dataseti, and ADataseti is the set of abnormal records in Dataseti. The threshold $Dt\ h = 1$ is generally set as a linear function of the number of terminals in the coverage region of the edge server, and the threshold on each edge server may be different.

③ *Triggered Update (TUpdate).* The edge server performs the local training when receiving abnormal log records from any terminal in its coverage region. The local fault detection model may be updated as soon as possible in this strategy, and the updated parameters will be sent to the cloud. It may lead to a large amount of data transmission between the edge servers and the cloud when there are many terminals in the power grid. This strategy can be seen as a special case of AUpdate with threshold $Dt\ h = 1$.

The above three strategies should be used in different situations; PUpdate and AUpdate can reduce the data transmission between edge servers and the cloud, while TUpdate can quickly update the fault detection model. We can choose different strategies according to the practical applications, and the strategies can be adopted on different edge servers simultaneously. The main difference of the three updating strategies is the updating frequency of the fault detection model; TUpdate may lead to high and unpredictable updating frequency, while the updating frequency of PUpdate and AUpdate can be controlled by changing parameters period and $Dt\ h$.

3.3. Model Updating. After receiving the parameters of local models from edge servers, the cloud will aggregate the parameters to update the global detection model and then send the updated parameters to all edge servers. Each edge server will further send the updated parameters to all terminals in its coverage region. Suppose the parameters of local models from k edge servers are the set $V = \{v_1, v_2, \ldots, v_k\}$, and the parameters of each local model are a vector $V_i = \langle v_1^i, v_2^i, \ldots, v_p^i \rangle$, where $1 \leq i \leq k$ and p is the number of parameters of the fault detection model. The set of parameters V is aggregated in the cloud, and we can get the new parameter vector $V_u = \langle v_1^u, v_2^u, \ldots, v_p^u \rangle$, in which each parameter is updated according to the following rules:

$$v_i^u = \eta . v_i^{'u} + (1 - \eta) \sum_{j=1}^{k} v_i^j, \tag{9}$$

where $v_i^{'u}$ is the original value of the ith global parameter, and $\eta$is the weighted coefficient and can be obtained by the cloud in the step of pretraining.

The accuracy of the fault detection model on the terminals depends on the frequency of model updating in the cloud, if the architecture of the grid power and the method to process the system log are given. The parameters of local models from the edge servers will be less when the cloud updates the global model frequently, which will increase the data transmission to send the updated parameters to all the edge servers and terminals. If the frequency of model updating is low, then some edge servers, on which the frequency of local training is high, will send their local parameters several times to the cloud, and only the latest version of the local parameters will be aggregated to update the global model, so that the influences of some faults will be ignored and the accuracy of the fault detection model will be affected.

Therefore, in order to improve the accuracy of the fault detection model, the cloud can adopt different strategies to update the model, such as PUpdate, AUpdate, and TUpdate. These update strategies in the cloud for model updating are similar to that on the edge server for local training, and the only difference is the measurement of the thresholds in each strategy. The update period of PUpdate in the cloud is typically set as the shortest time of local training on all edge servers. The threshold of AUpdate in the cloud will be set according to the number of edge servers, which have been finished the local training, that is, the number of parameter vectors received by the cloud. The cloud will update the model when receiving parameters of local model from any edge server when TUpdate is used in the cloud. The update strategy in the cloud is determined by the experts according to the requirements and configurations of the power grid and the strategies adopted on the edge servers.

3.4. Fault Detection Algorithm. Suppose there are one cloud Cloud and k edge servers $\{\text{Server}_1, \text{Server}_2, \ldots, \text{Server}_k\}$ in the power grid, and the terminals in the coverage region of edge server Server_i are $\{de\ v_{i_1}, de\ v_{i_2}, \ldots, de\ v_{i_n}\}$, then the fault detection process is as follows:

Step 1. The historical system logs Records are used to pretrain the initial fault detection model using LSTM in Cloud. Then, Cloud sends the parameter vector V_{init} to all edge servers $\{\text{Server}_1, \text{Server}_2, \ldots, \text{Server}_k\}$, and each edge server Server_i further sends V_{init} to the h terminals $\{de\ v_{i_1}, de\ v_{i_2}, \ldots, de\ v_{i_n}\}$ in its coverage region.

Step 2. When a log record r_i is generated on a terminal $de\ v_{i_j}$, it should be firstly preprocessed by variable replacement and input into the fault detection model to check whether a fault occurs and determine the type of the fault on dev_{i_j}. Then, the log records within a time period will be compressed and sent to the edge server $Server_i$.

Step 3. Each edge server $Server_i$ collects the compressed system logs $R_i = \{R_{i_1}, R_{i_2}, \ldots, R_{i_h}\}$ from terminals $\{de\ v_{i_1}, de\ v_{i_2}, \ldots, de\ v_{i_n}\}$ in its coverage region and trains the local LSTM model using dataset R_i according to the updating strategy, such as PUpdate, AUpdate, or TUpdate. Once completing the local training, $Server_i$ transmits the parameters V_i of local model to Cloud.

Step 4. After receiving the parameters $V = \{V_1, V_2, \ldots, V_k\}$ from edge servers $\{Server_1, Server_2, \ldots, Server_k\}$, Cloud aggregates V to generate the new parameters $V_{update} = \langle v_1^u, v_2^u, \ldots, v_p^u \rangle$ using PUpdate, AUpdate, or TUpdate strategy. Then, V_{update} will be transmitted to each edge server $Server_i$, and finally to all terminals $\{de\ v_{i_1}, de\ v_{i_2}, \ldots, de\ v_{i_n}\}$, and steps 2–4 will continue until the fault detection is interrupted.

In the above algorithm, each terminal continuously generates a large amount of system log, and each record in the system log is input into the fault detection model to find as many faults as possible. The system log will be compressed before transmitting to the edge server, and only the parameters of local model are sent to the cloud to reduce the transmission delay. Meanwhile, the fault detection on the terminal, the local model training on edge server, and the model updating in the cloud can fully utilize the computing resources of different hardware. Therefore, high accuracy, low latency, less data transmission, and privacy protection are taken into consideration in this federated learning-based fault detection algorithm.

4. Performance Evaluation

The fault detection algorithm proposed in this study is evaluated in a simulated power grid with a cloud, 10 edge servers, and a number of terminals. The terminal is configured as a brand of energy controller, the system log is collected in practical application, and the summary of the system log is listed in Table 3. The time of model updating, the amount of data transmission of each terminal, and the fault detection accuracy are tested to evaluate our fault detection algorithm. The weights in the similarity of records are set as $\alpha = \beta = 0.5$.

The average update time of the model using different update strategies when the number of terminals in the coverage region of the edge server changes is shown in Figure 4. It can be seen that the update time of PUpdate strategy keeps stable if the update period is fixed, while the update time of AUpdate and TUpdate strategies decreases with the increase of terminals. The reason is that the increase terminals lead to high frequency of the edge server receiving abnormal log records, and the local update will be performed more frequently if the other conditions is fixed. Since the threshold D_{th} in AUpdate is set to be 10, and the TUpdate can be seen as a special case of AUpdate with $D_{th} = 1$, the average update time of TUpdate is significantly lower than that of AUpdate strategy for the same number of terminals.

Table 3: Summary of system log generated by an energy controller within an hour.

No.	Information	Value
1	Number of log records	1019
2	Average number of words per record	9.25
3	Maximum frequency of the words	880
4	Minimum frequency of the words	1
5	Size of raw system log	70 KB
6	Size of compressed system log	18 KB

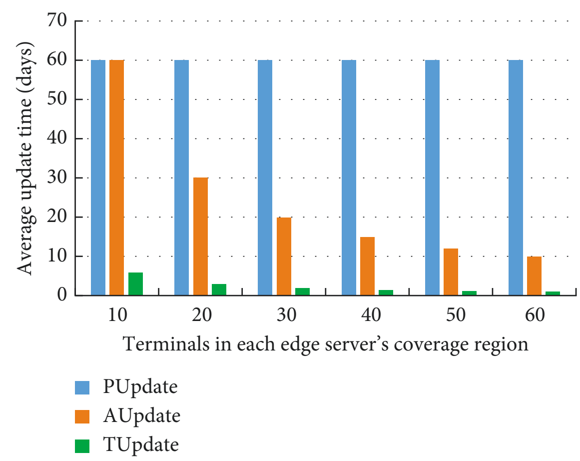

Figure 4: Average update time of the fault detection model.

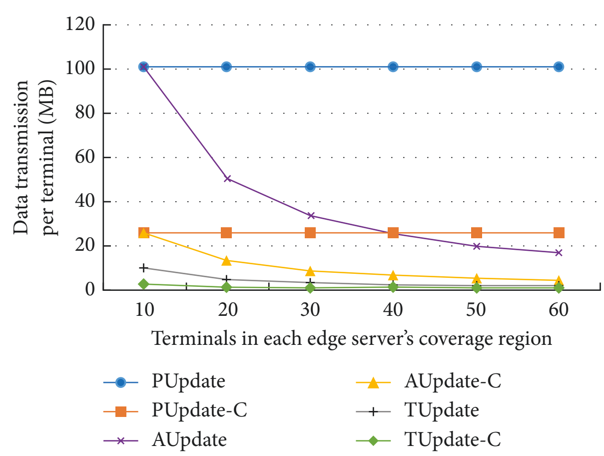

Figure 5: Average data transmission of a power terminal in the fault detection model.

The average data transmission of each terminal for the local training on edge server with different update strategies is shown in Figure 5. The legends "X" and "X-C" represent the two cases when the terminal transmits raw system log and compressed system log to edge server with update strategy X respectively. It can be seen from Figure 5 that log compression can significantly reduce the data transmission in all update strategies, and PUpdate generates the largest data transmission, followed by AUpdate, and data transmission in TUpdate is the smallest. The main reason is that PUpdate takes the longest period for each local training, while the average time of local training in TUpdate is the

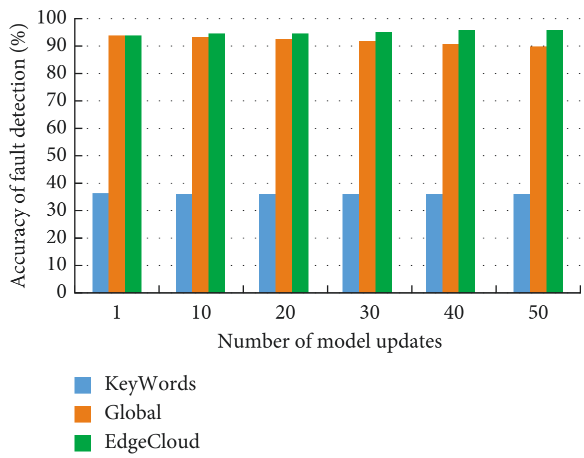

Figure 6: Accuracy of the fault detection results.

shortest. Meanwhile, when the number of terminal devices increases, the model update time of AUpdate and TUpdate gradually approaches. In the case of log compression, the data transmission of AUpdate and TUpdate approximates to be equal.

From the above results, it can be seen that the PUpdate strategy has the longest update period and the largest amount of data transmission for each local training. The TUpdate has the shortest update period and the least data transmission for each local training. The update period and data transmission in AUpdate are less than that of PUdate and more than that of TUpdate. In practical applications, TUpdate is suitable for newly deployed terminals and terminals with high security requirements, so as to quickly collect the abnormal system records of the terminal and train a more accurate fault detection model. Terminals with limited network bandwidth can also avoid network congestion and packet loss caused by a large amount of data transmission using TUpdate strategy. PUpdate is applicable to terminals with stable operation, less strict security requirements, and large network bandwidth, and the update period can be set according to the actual application and is usually initialized as the average failure time of the terminal. Strategy AUpdate has the maximum flexibility, the update period can be changed easily by modify the threshold D_{th}, and the update period grows as D_{th} become larger.

The accuracy of fault detection using different fault detection methods is shown in Figure 6. The legends "KeyWords" is the keyword matching-based method, "Global" is the global LSTM model, that is, the fault detection model is trained in the cloud and cannot be updated, and "EdgeCloud" is our LSTM model with edge and cloud cooperation. The results show that the accuracy of Keywords method is very low since the abnormal behavior of some faults does not contain the keywords. The accuracy of both Global and EdgeCloud methods is obviously higher than KeyWords because the faults are recognized by the natural language processing method LSTM. With the accumulation of the system log, the Global method does not update the fault detection model, which leads to decreasing of the accuracy of fault detection. The new system log is used to train the local model and further update the global fault detection model to increase the accuracy of the fault detection model. With the increase of the number of model updates in the cloud in EdgeCloud, its accuracy becomes higher than that of Global method.

5. Conclusions

In this study, we analyze the characteristics of power terminal fault detection in the power grid and propose a three-tier fault detection model based on federated learning. The accuracy of fault identification and the privacy protection of terminal are both taken into consideration in this model, then the model training, model updating, and fault detection are performed at different levels. Log compression and transmission of parameters of the model are used to reduce data transmission and protect the privacy, the LSTM model is used to improve fault detection accuracy, and three different model update strategies are used to further improve the accuracy of fault detection. The experimental results show that the log compression method can effectively reduce the amount of data transmission. The three model strategies are suitable for different application scenarios and terminals, and the detection accuracy of our proposed fault detection model is higher than that of the traditional keyword-based models and the global models based on historical data. In the future work, it is planned to integrate our proposed fault detection model into various embedded operating systems and deploy it on different devices.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the Science and Technology Project of State Grid Corporation of China (grant no. 5700-202055484A-0-0-00).

References

[1] J. Park, S. Kim, J. Lee, J. Ham, and K. Oh, "Automatic inspection drone with deep learning-based auto-tracking camera gimbal to detect defects in power lines," in *Proceedings of the International Conference on Vision, Image and Signal Processing*, pp. 1–46, Vancouver, Canada, August 2019.

[2] H. Bouazza, M. L. Bendaas, T. Allaoui, and M. Denai, "Application of artificial intelligence to wind power generation: modelling, control and fault detection," *International Journal of Intelligent Systems Technologies and Applications*, vol. 19, no. 3, pp. 280–305, 2020.

[3] L. Shi, P. Yu, Z. Zhou et al., "DC-DC power supply fault prediction and analysis based on monitoring parameter simulation and LSTM network model," in *Proceedings of the*

International Conference on Algorithms, Computing and Artificial Intelligence, pp. 1–8, Sanya, China, December 2021.

[4] N. C. Phuong, "Large power transformer overload detection using sound analysis," in *Proceedings of the International Conference on Signal Processing and Machine Learning*, pp. 135–141, Beijing China, August 2021.

[5] J. S. Hong, S. Y. Hyun, Y. W. Lee, J. H. Choi, S. J. Ahn, and S. Y. Yun, "Detection of open conductor fault using multiple measurement factors of feeder RTUs in power distribution networks with DGs," *IEEE Access*, vol. 9, no. 9, pp. 143564–143579, 2021.

[6] Y. Li, F. Yu, Q. Cai et al., "Image fusion of fault detection in power system based on deep learning," *Cluster Computing*, vol. 22, no. 4, pp. 9435–9443, 2019.

[7] T. Yu, M. Xie, X. Li, Y. Ling, D. Bin, and C. Yang, "Network attack detection method for power system terminal based on improved random forest," in *Proceedings of the International Conference on Artificial Intelligence and Information Systems*, vol. 85, pp. 1–85, Chongqing, China, May 2021.

[8] Q. Sun, X. Yu, H. Li, F. Peng, and G. Sun, "Fault detection for power electronic converters based on continuous wavelet transform and convolution neural network," *Journal of Intelligent and Fuzzy Systems*, vol. 42, no. 4, pp. 3537–3549, 2022.

[9] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[10] S. Weisong, S. Hui, C. Jie, Z. Quan, and L. Wei, "Edge computing: a new computing model in the era of everything," *Journal of Computer Research and Development*, vol. 54, no. 5, pp. 907–924, 2017.

[11] W. Zhang, J. Wang, G. Han, S. Huang, Y. Feng, and L. Shu, "A data set accuracy weighted random forest algorithm for IoT fault detection based on edge computing and blockchain," *IEEE Internet of Things Journal*, vol. 8, no. 4, pp. 2354–2363, 2021.

[12] T. T. Huong, P. B. Bac, D. M. Long et al., "LocKedge: low-complexity cyberattack detection in IoT edge computing," *IEEE Access*, vol. 9, no. 9, pp. 29696–29710, 2021.

[13] A. Mishra, A. Cohen, T. Reichherzer, and N. Wilde, "Detection of data anomalies at the edge of pervasive IoT systems," *Computing*, vol. 103, no. 8, pp. 1657–1675, 2021.

[14] A. Mudgerikar, P. Sharma, and E. Bertino, "Edge-based intrusion detection for IoT devices," *ACM Transactions on Management Information Systems*, vol. 11, no. 4, pp. 1–18, 2020.

[15] W. Huo, F. Liu, L. Wang, Y. Jin, and L. Wang, "Research on distributed power distribution fault detection based on edge computing," *IEEE Access*, vol. 8, no. 8, pp. 24643–24652, 2020.

[16] Y. Yin, J. Lin, N. Sun et al., "Method for detection of unsafe actions in power field based on edge computing architecture," *Journal of Cloud Computing*, vol. 10, no. 1, pp. 1–17, 2021.

[17] Y. Yang, J. Mao, Y. Chen, H. Zhong, Z. Huang, and Y. Wang, "Semi-supervised cloud edge collaborative power transmission line insulator anomaly detection framework," *Lecture Notes in Computer Science*, in *Proceedings of the International Conference on Image and Graphics*, pp. 210–221, Haikou, China, August 2021.

[18] S. Zhang, J. Wang, J. Tong, J. Zhang, and M. Zhang, "Cloud-edge fusion based abnormal object detection of power transmission lines using incremental learning," *IEEE Access*, vol. 8, no. 8, pp. 218694–218701, 2020.

[19] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pp. 1273–1282, Florida, FL, USA, April 2017.

[20] M. Wen, R. Xie, K. Lu, L. Wang, and K. Zhang, "FedDetect: a novel privacy-preserving federated learning framework for energy theft detection in smart grid," *IEEE Internet of Things Journal*, vol. 9, no. 8, pp. 6069–6080, 2022.

[21] J. Lin, J. Ma, and J. Zhu, "Privacy-preserving household characteristic identification with federated learning method," *IEEE Transactions on Smart Grid*, vol. 13, no. 2, pp. 1088–1099, 2022.

[22] Z. Su, Y. Wang, T. H. Luan et al., "Secure and efficient federated learning for smart grid with edge-cloud collaboration," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 2, pp. 1333–1344, 2022.

[23] Y. Wang, I. L. Bennani, X. Liu, M. Sun, and Y. Zhou, "Electricity consumer characteristics identification: a federated learning approach," *IEEE Transactions on Smart Grid*, vol. 12, no. 4, pp. 3637–3647, 2021.

[24] Q. Liu, B. Yang, Z. Wang et al., "Asynchronous decentralized federated learning for collaborative fault diagnosis of PV stations," *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 3, pp. 1680–1696, 2022.