Blockchain-Based Federated Learning for Device Failure Detection in Industrial IoT

Weishan Zhang, *Member, IEEE*, Qinghua Lu, *Senior Member, IEEE*, Qiuyu Yu, Zhaotong Li, Yue Liu, Sin Kit Lo, Shiping Chen, *Senior Member, IEEE,* Xiwei Xu, *Member, IEEE*, and Liming Zhu

***Abstract*—Device failure detection is one of most essential problems in Industrial Internet of Things (IIoT). However, in conventional IIoT device failure detection, client devices need to upload raw data to the central server for model training, which might lead to disclosure of sensitive business data. Therefore, in this article, to ensure client data privacy, we propose a blockchain-based federated learning approach for device failure detection in IIoT. First, we present a platform architecture of blockchain-based federated learning systems for failure detection in IIoT, which enables verifiable integrity of client data. In the architecture, each client periodically creates a Merkle tree in which each leaf node represents a client data record, and stores the tree root on a blockchain. Furthermore, to address the data heterogeneity issue in IIoT failure detection, we propose a novel centroid distance weighted federated averaging (CDW_FedAvg) algorithm taking into account the distance between positive class and negative class of each client data set. In addition, to motivate clients to participate in federated learning, a smart contact-based incentive mechanism is designed depending on the size and the centroid distance of client data used in local model training. A prototype of the proposed architecture is implemented with our industry partner, and evaluated in terms of feasibility, accuracy, and performance. The results show that the approach is feasible, and has satisfactory accuracy and performance.**

***Index Terms*—AI, blockchain, edge computing, failure detection, federated learning, IoT, machine learning.**

I. Introduction

DEVICE failure detection is one of the most essential problems in Industrial Internet of Things (IIoT) [1], [2]. In conventional device failure detection of IIoT, client devices need to upload local raw data to the central server for centralized model training. This leads to the issue of data privacy as clients' local data might be business sensitive. For example, the hotels' air-conditioning usage data might reflect occupancy rate.

Federated learning is an emerging machine learning paradigm [3], [4] in a way that preserves privacy and reduces bias in model training. In each round of federated learning, multiple clients (e.g., organizations, data centers, IoT, or mobile devices) are selected to train models locally to produce a global model while raw data are stored in clients and not exchanged or transferred.

Blockchain has been recently leveraged in IIoT federated learning to provide data integrity and incentives to attract sufficient client data and computation resources for training [5]–[7]. However, there is a lack of systematic and holistic architecture design to support methodical development and efficient methods to tackle the challenge of data heterogeneity in device failure detection of IIoT.

Therefore, in this article, we present a blockchain-based federated learning approach for device failure detection in IIoT. To produce an unbiased global model, the model updates are aggregated using a novel averaging algorithm called centroid distance weighted federated averaging (CDW_FedAvg), which takes into account the distance between the positive class and the negative class of each client data set in weight calculation. In this study, "positive class" means the detected failures while "negative class" indicates the normal operations of IoT devices. Client data are hashed and stored periodically on blockchain to ensure data integrity while addressing the performance issue of blockchain. Incentives are calculated based on clients' data contributions (i.e., the size and centroid distance of client data) and rewarded to motivate the clients via a smart contract. A proof-of-concept prototype of the proposed architecture is implemented with our industry partner and evaluated in terms of feasibility, accuracy, and performance. The evaluation results show that the proposed architecture is feasible and has satisfactory detection accuracy and performance.

The contributions of this article are as follows.

1) A platform architecture that provides a systematic view of system interactions and serves as a guidance for the design of the blockchain-based federated learning systems in IIoT failure detection. The architecture design involves the following architectural design decisions: placement of model training, storage of monitored client data, incentive mechanism for clients, and deployment of blockchain.

Manuscript received August 19, 2020; revised October 9, 2020; accepted October 17, 2020. Date of publication October 20, 2020; date of current version March 24, 2021. This work was supported in part by the National Natural Science Foundation of China under Grant 62072469; in part by the National Key Research and Development Program under Grant 2018YFE0116700; in part by the Shandong Provincial Natural Science Foundation (Parallel Data Driven Fault Prediction under Online-Offline Combined Cloud Computing Environment), under Grant ZR2019MF049; and in part by the Fundamental Research Funds for the Central Universities under Grant 2015020031. *(Corresponding author: Qinghua Lu.)*

Weishan Zhang, Qiuyu Yu, and Zhaotong Li are with the College of Computer Science and Technology, China University of Petroleum (East China), Qingdao 266580, China.

Qinghua Lu, Shiping Chen, Xiwei Xu, and Liming Zhu are with Data61, CSIRO, Sydney, NSW 2015, Australia, and also with the School of Computer Science and Engineering, University of New South Wales, Sydney, NSW 2052, Australia (e-mail: qinghua.lu@data61.csiro.au).

Yue Liu and Sin Kit Lo are with the School of Computer Science and Engineering, University of New South Wales, Sydney, NSW 2052, Australia, and also with Data61, CSIRO, Sydney, NSW 2015, Australia.

Digital Object Identifier 10.1109/JIOT.2020.3032544

2) A federated averaging algorithm called CDW_FedAvg taking into account the distance between the positive class and the negative class of each client data set to reduce the impact of the data heterogeneity issue in IIoT device failure detection.
3) A blockchain anchoring protocol that builds a custom Merkle tree in which each leaf node represents a record of data collected by a client's edge device, and places the root of the custom Merkle tree at a preconfigured interval on blockchain to ensure verifiable integrity of client data in an efficient way.
4) An incentive mechanism via the designed incentive smart contract to motivate clients to contribute their data and computational resources for model training. The tokens rewarded to each client are dependent on the size and the centroid distance between the positive class and the negative class of data contributed to model training.
5) Feasibility, detection accuracy, and performance evaluation using a real industry use case that detects failures of water-cooled magnetic levitation chillers in air-conditioners.

The remainder of this article is organized as follows. Section II summarizes the related works. Section III presents the proposed solutions. Section IV discusses the implementation and evaluation. Section V concludes this article and outlines the future work.

II. Related Work

IoT has been integrated into industrial systems to allow a higher degree of automation and improve the economic benefits, including productivity and efficiency. In industrial IoT, traditional centralized cloud-based computing approach may be inefficient, since tasks are pushed to the cloud from numerous IoT devices that may cause data leakage and network overload [8]. To address the issue of data privacy and limited bandwidth in cloud computing, edge computing has been widely applied in IIoT systems to shift part of computing tasks from cloud servers to edge nodes, which are closer to where data are generated.

The concept of federated learning is first proposed by McMahan *et al.* [9], which enables edge computing from the machine learning perspective. In a border definition of federated learning, multiple clients are coordinated to train model locally and aggregate model updates to achieve the learning objective, while raw data are stored in clients and not exchanged or transferred [4], [10]. Compared with most studies of federated learning with central server, a few research works [5], [11] build a federated learning framework without central server, such as applying p2p structure.

Federated learning is classified into two types: 1) cross-device federated learning and 2) cross-silo federated learning. The former is introduced on mobile and edge device applications. For example, Google has applied federated learning setting to the prediction of search suggestions, next words and emojis, and the learning of out-of-vocabulary words [12]–[14]. The latter represents another type of federated learning setting in which clients are different organizations or geo-distributed datacenters [10]. For example, Sheller *et al.* [15] adopted cross-silo federated learning to build a segmentation model on the brain tumor data from multiple institutions.

There have been dramatically surged numbers of studies conducted to address statistical heterogeneity of federated learning. Nishio and Yonetani [16] designed a novelty federated learning protocol with consideration of heterogeneous client properties to reduce the impact of skewed data. Han and Zhang [17] discussed to cope with the negative impact of the systematic data corruption via collaborative machine teaching in FL. Pang *et al.* [18] proposed a deep deterministic policy gradient-based method for updates aggregation to deal with the issue of statistical heterogeneity in IoT. However, the effect of centroid distance of data set is not considered in the above studies, which may cause influence to the overall performance.

Blockchain was first introduced as the technology supporting Bitcoin [19], and has been generalized to an append-only data store that is distributed across computational nodes and structured as a linked list of blocks containing a set of transactions [20], [21]. Currently, many studies have integrated blockchain into their federated learning approaches. Since federated learning relies on a single server, which is vulnerable to malfunction, some studies are focused on eliminating single-point failure in federated learning using blockchain [5], [11]. Zhao *et al.* [11] designed a system based on federated learning and blockchain to let clients empower full nodes and compete to serve as a central server by turns for aggregation. Kim *et al.* [5] proposed a blockchained-federated learning architecture in which participant clients are able to store local model updates on blocks and all clients as miners can access and aggregate the updates through smart contracts, without a single central server. Some works leverage blockchain in federated learning for auditability and data privacy [7], [22]. Lu *et al.* [22] incorporated federated learning and blockchain into a data sharing scheme to preserve privacy. Majeed and Hong [7] recorded local model parameters in each iteration into blocks, and introduces "the global model state trie" to securely store global model. Incentive mechanism is another important research direction, which is imported to fedearted learning to encourage participants and standardize the behavior of participants [6], [23]. Weng *et al.* [6] employed incentive mechanism and blockchain transactions to protect the privacy of local gradients, and enable auditability of the training process. Kang *et al.* [23] proposed an incentive mechanism based on blockchain combing the reputation which is introduced to measure the trustworthiness and reliability of clients, to stimulate honest clients with high-quality data to participate in federated learning. Moreover, there are some recently proposed platforms that increase more novel participant roles, such as buyers who can pay for the clients in order to complete their training tasks [24]. Bao *et al.* [24] provided a platform for buying federated learning models and designed an architecture to reduce the time cost of buyers querying and verifying models on a blockchain.

However, the above studies do not present the architecture design details of blockchain-based federated learning systems with consideration of the data heterogeneity issue in IIoT

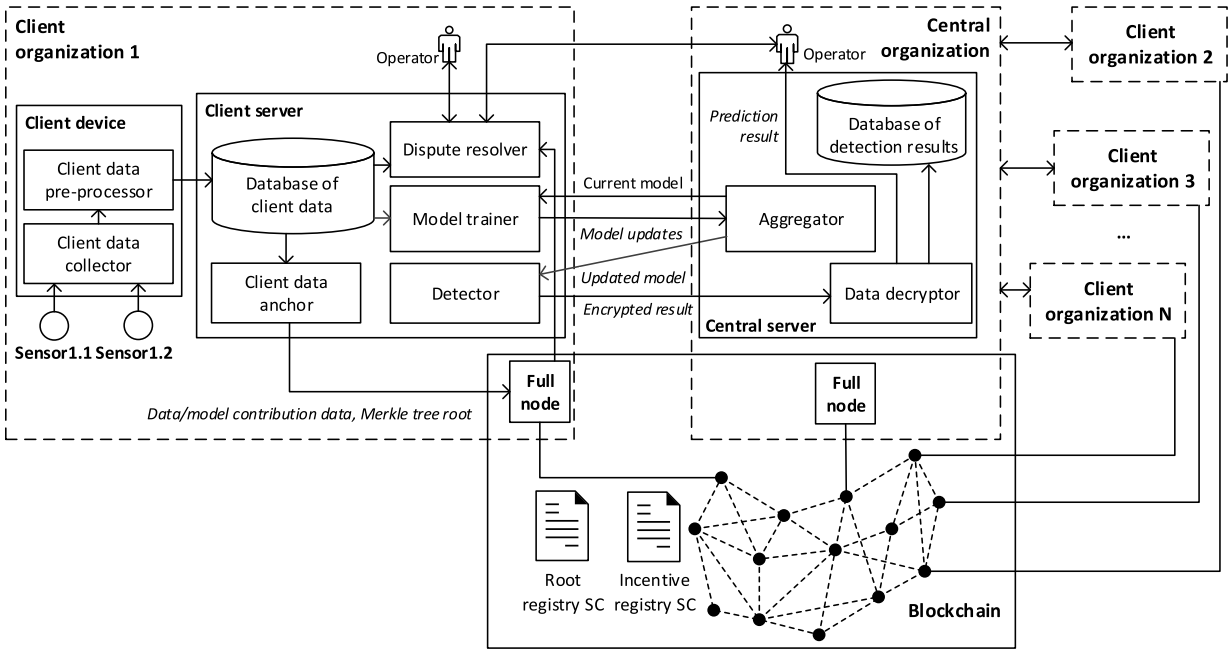

Fig. 1. Architecture of blockchain-based federated learning systems for failure detection in IIoT.

failure detection. Additionally, how to ensure the integrity of client data is not discussed. In this article, we propose an architecture of federated learning for IIoT failure detection using blockchain in a way that motivates clients to provide sufficient data and computing resources for model training while reducing the impact of data heterogeneity and preserving client data privacy and integrity.

III. Approach

We adopt the standard requirements elicitation methodology [25] for our industry partner's specific requirements and derive a list of general application-agnostic functional and nonfunctional requirements for blockchain-based federated learning systems for IIoT device failure detection. The gathered functional requirements include: 1) making detection using a trained model; 2) resolving disputes between platform owner and clients; and 3) rewarding clients for their contributions to the model training process. The identified nonfunctional requirements include: 1) preserving edge data privacy; 2) maintaining satisfactory detection accuracy; 3) ensuring edge data integrity; and 4) keeping availability of blockchain infrastructure.

In this section, we present a platform architecture for blockchain-based federated learning systems to meet the above general application-independent requirements for federated learning systems. Section III-A provides an overview of the architecture and discusses how the components and their interactions. Section III-B discusses the main architectural decisions. Section III-C proposes a novel federated averaging algorithm called CWD_FedAvg. Section III-D presents a protocol for anchoring the client data to the blockchain to ensure data integrity. Section III-E designs an incentive mechanism to encourage the clients to contribute to model training.

A. Overall Architecture

Fig. 1 illustrates the overall platform architecture, which consists of two types of participating organizations: 1) central organization and 2) client organization. The central organization is the federated learning platform owner (e.g., manufacturer company) that maintains all industrial services based on detection results analyzed by the platform, while the client organizations are the organizations that contribute local data and computation resources for local model training.

Each client organization maintains sensors, a client device, a client server, and a blockchain full node, while the central organization hosts a central server and a blockchain full node. Each client device (e.g., Raspberry Pi) gathers the environment data detected by sensors through *client data collector* and clean the data (e.g., noise reduction) via *client data preprocessor*. All the collected client environment data are stored in *database of client data* hosted by the client server. If a client organization is selected by the central server for a certain round of model training, the client server downloads the current version of the global model from the central server to further train the model and compute model updates using the local data via *model trainer*. The model updates are sent to the *aggregator* on the central server at a preconfigured round. *Aggregator* combines the model updates received from the clients and produces a global model, which is then sent back to client servers for *detector*. The detection results are encrypted and sent to the central organization, where the operator can decrypt the received detection results via *data decryptor* and make the final decision.

Each client server periodically creates a Merkle tree in which each leaf node represents a data record collected by sensors. The information of a client in each selection round (including the Merkle tree root, the status of training, and the size and the centroid distance of client data for model

training) are all stored in the predeployed smart contracts on the blockchain through *client data anchor*. If a dispute (e.g., about failure cause) occurs, the operators of both the central organization and the client organization can use *dispute resolver* to verify the integrity of client historical data in a certain anchoring period, by comparing the data with the respective Merkle tree root stored on-chain. To reward the client organizations for their contribution to model training, tokens are distributed to them according to the contributed size and centroid distance of client data used for model training.

B. Architectural Design Decisions

1) Local Model Training: To preserve the privacy of client data (e.g., usage frequency and time), which is often sensitive to each client organization's business, the architecture adopts federated learning to train the models locally on client servers via *model trainer* while computing training results on the central server to formulate a global model for detection via *aggregator*.

2) Anchoring Monitored Data to Blockchain: To resolve disputes between the central organization and client organizations (e.g., about failure causes), the architecture leverages the blockchain to enable verifiable integrity of client data via *client data anchor* on the client server. Since blockchain has limited storage capabilities, the client server periodically creates a Merkle tree in which each leaf node represents a record of data collected by sensors, and stores its root in the predeployed root registry smart contract (SC) through *client data anchor*.

3) Smart Contract-Based Incentive Mechanism: The architecture provides incentives to encourage client organizations for model training. Tokens are rewarded to client organizations based on the size and the centroid distance of data taken in model training. The rewarded tokens are recorded in *incentive registry SC*.

4) Blockchain Deployment: In a blockchain network, a full node maintains a complete list of every single transaction that had occurred on the blockchain. In the proposed architecture, each participating organization, including central organization and client organization, hosts one blockchain full node. Hence, each organization has a full replica of the data stored on the blockchain that can be used for auditing, and ensure the availability of the whole system.

C. Centroid Distance Weighted Federated Averaging

The federated averaging (FedAvg) algorithm proposed by Google [3] is not suitable for industrial IoT systems since the heterogeneity between the different client local data sets exists heavily in industrial IoT systems (i.e., the distribution and size of the data set on each client may be different). Take the air-conditioner failure detection as an example, the deployed air-conditioners might have different working environments (e.g., weather) and different users (e.g., age).

To reduce the impact of data heterogeneity and improve the model performance, we propose a CDW_FedAvg algorithm taking into account the distance between positive class and negative class of each client data set. The weighted average process are calculated as

$$w_t = \frac{\sum_{k=1}^{K} s_t^k * f\left(d_t^k\right) * w_t}{\sum_{k=1}^{K} s_t^k * f\left(d_t^k\right)\Big)}, \quad h\left(d_t^k\right) = 1/d_t^k \tag{1}$$

where s_t^k represents the size of *k_th* client training data set contributed in *t_th* round. d_t^k denotes the distance between the centroid of positive class and the centroid of negative class from the *k_th* client data set in *t_th* round. $f(d_t^k)$ is the function to process d_t^k, outputting the reciprocal of d_t^k. Here, we sacrifice the model accuracy for client data sets that have greater distance between the centroid of positive class and centroid of negative class since most of the client data sets in the real-world have smaller distance. The detailed process is illustrated in Algorithm 1.

Algorithm 1 CWD_FedAvg Algorithm

```
/*Central server*/
Initialize w_0
for each round t = 1, 2, ... do
    Select K clients
    for each client k ∈ K clients do
        w_t^k ← UpdateClient( )
        d_t^k ← the distance between two classes in training dataset
    end for
    f(d_t^k) ← 1/d_t^k
    w_t ← Σ_{k=1}^{K} n_k * f(d_t^k) * w_{t-1} / Σ_{k=1}^{K} n_k * f(d_t^k)
end for
/*Client update*/
UpdateClient( ){
Initialize local minibatch size B, local epochs E, learning rate η
for each epoch i ∈ E do
    randomly sample S_i based on size B
    w_i ← w_{i-1} - η∇g(w_{i-1} ; S_i )
end for
return w_i
}
```

In Algorithm 1, for each round, the central server selects a subset of clients to participate in the round based on some predefined criteria (e.g., idle and charging) and sets the local minibatch size B, local epochs E, and learning rate η on the selected clients. Then, the selected clients execute local stochastic gradient descent (SGD) on B batches in E epochs, calculate the distance d_t^k, and return the update w_t and d_t^k to the central server. Finally, on the central server, the updates are weighted averaged according to the size of local training data set and the reciprocal of d_t^k.

D. Anchoring Protocol

An anchoring protocol is designed to build a Merkle tree based on the monitored data collected by a client, and anchor the Merkel tree root to the blockchain for client data integrity and auditing (i.e., dispute resolution between the central organization and the client organization). The incentive mechanism is also supported by the anchoring protocol to reward the client organizations according to the anchored contribution recorded on the *incentive registry SC*.

Algorithm 2 describes how the anchoring protocol works. The anchoring protocol is scheduled according to the agreement between the central organization and client organizations (e.g., every hour). For each anchoring period p_i, the protocol

Algorithm 2 Anchoring Protocol

```
for each period p_i = 1, 2, ... do
    for c_j ∈ C do
        for d^c_pn do
            d^c_pn ← String( d^c_pn)
            d^c_pn ← Hash( d^c_pn)
            pNodeList[0..N] ← d^c_pn
        end for
        repeat
            for (k = 0 ; k < length(pNodeList) ; k+2) do
                l ← pNodeList[k]
                if k = length(pNodeList) then
                    r ← l
                else
                    r ← pNodeList[k + 1]
                end if
                r ← SHA256(l + r)
                temp[j] ← r
                (each j ∈ [0,⌈k/2⌉])
            end for
            pNodeList[0..j] ← temp[0..j]
        until length(pNodeList) = 1
        root ← pNodeList[0]
        time ← current time
        UpdRootSC(time, root)
    end for
end for
for each training round do
    model training
    UpdStaSC(rNo, dataSize, distance)
    CalIncentiveSC( )
end for
```

goes through each client organization c_j, and queries the client data d^c_{pn} collected within the anchoring period. The protocol converts each data record d^c_{pn} to a string format. All the data collected within the anchoring period are hashed and used to construct a Merkle tree where each leaf node represents the hash value of a data record.

To construct the Merkle tree structure, the protocol first creates a *pNodeList* to store the parent node values and a *temp* to put the results of the combined hash values of the two child nodes. All the leaf node values are initially placed in the *pNodeList*. To compute the parent node of each pair of two leaf nodes that are adjacent to each other, the hash values of them are assigned to l which denotes left child node, and r which denotes right child node, respectively, and combined to produce the hash value for the parent node, which is then added to *temp*. After going through all the elements in *pNodeList*, the *temp* is converted to *pNodeList*. The tree construction process is executed recursively until the length of the *pNodeList* becomes one. The root is exactly the only value in the *pNodeList*.

UpdRootSC() is used to anchor the Merkle tree *root* and timestamp of the current period to *root registry SC*. Once the client servers finish their work and sent the model updates to the central server, *dataSize* (the size of data for training) and *distance* (the centroid distance between two classes of training data set) of each participant client server and the serial number of training round *rNo* are anchored in the blockchain by *UpdStaSC*(). The corresponding tokens are then calculated and rewarded to client organizations through *CalIncentiveSC*(). Both *UpdStaSC*() and *CalIncentiveSC*() are encoded in Incentive registry SC.

Algorithm 3 Incentive Calculation and Distribution

```
/*Central server executes*/
Clist ← selected clients address
/*Client server executes*/
struct {finished, dataSize, distance} training
for each training round do
    if address ∈ Clist then
        model traning
        UpdStaSC(rNo, dataSize, distance)
        CalIncentiveSC( )
    end if
end for
/*Update Status of Training work */
UpdStaSC(rNo, dataSize, distance){
    training.finished ← true
    training.dataSize ← dataSize
    training.distance ← distance
    contri[addr][rNo] ← work
}
/*Calculation Incentive*/
CalIncentiveSC( ){
if address ∈ Clist then
    if contri[addr][rNo].finished == true then
        token[addr] = token[addr] + contri[addr][rNo].dataSize
                    + contri[addr][rNo].distance * C
    end if
end if
}
```

E. Incentive Mechanism

Algorithm 3 describes how the incentive mechanism works. The incentive mechanism aims to reward client servers with tokens, according to their contribution (the size of data applied for model training and the distance between two classes of training data set). For every training round, the central server chooses a fixed number of client servers, and stores their blockchain addresses in *Clist*. Besides, the central server should also define a struct *training* to enroll finished training tasks of client servers, including the status of training work *finished*, the size of data for training *dataSize*, and the centroid distance between two classes of training data set *distance*. The chosen client servers train models locally, and upload corresponding information into *training*, which is then converted to *contri*. Finally, the incentive *tokens* are distributed to each *address* of the client server that *Clist* records, in accordance with their *contri*, respectively. The number of tokens is the sum of the *dataSize* and a value computed by multiplying *distance* by a constant C. The ultimate goal of this incentive mechanism is to encourage more client servers for participation, to eliminate the model bias.

IV. Implementation and Evaluation

In this section, we implement a proof-of-concept prototype using a real-world use case, and evaluate the prototype in terms of feasibility accuracy and performance.

A. Use Case

Our industry partner is one of the largest air-conditioner manufacturers in the world, and a large portion of their clients are hotels that are distributed across the globe. The failures of the hotel air-conditioning systems directly influence customer satisfaction. In the collaborated project, we aim to predict

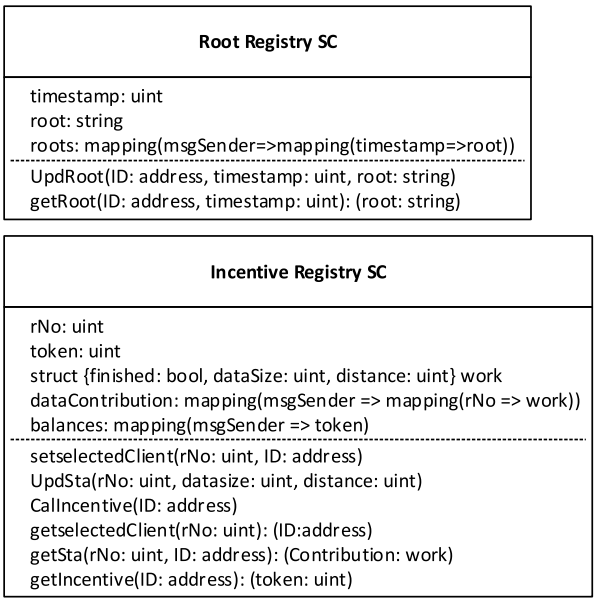

Fig. 2. On-chain data structure.

failures for air-conditioning systems. In the use case, we claim that the manufacturer owns the servers/devices placed in the hotels. The ownership of hotel servers/devices here means only the manufacturer has the write-access authority to the firmware/software on the devices/servers.

In particular, the project seeks to place most of the machine learning tasks within the hotels to preserve the data privacy of hotels. For example, room occupancy percentage is sensitive to a hotel and can be calculated based on the air-conditioning system data. Also, the project aims to encourage hotels to contribute their air-conditioner data and train failure detection models locally. Tokens are rewarded to hotels based on their contribution that can later be used to buy new products or services.

The water-cooled magnetic levitation chiller is the key component in a hotel air-conditioning systems. We developed a proof-of-concept prototype for the failure detection system of water-cooled magnetic levitation chillers using the proposed architecture.

B. Implementation

Fig. 2 shows the design of two on-chain smart contracts for client data anchoring and model training incentives: *root registry SC* stores Merkle tree roots and corresponding timestamps, while *incentive registry SC* maintains incentive information, including training round number, the status of training task, size of client data, distance between two classes of client data applied for training work, and rewarded tokens based on data size and distance.

We adopt a federated learning framework named *Leaf* [26] in the implementation. We use Ethereum [27] as the underlying blockchain network, in which the consensus algorithm is Proof of Work (PoW), and the difficulty is configured to 0×4000. The smart contracts are written in Solidity and compiled with Solidity compiler version 0.4.20. The *client data anchor* component is developed in Java 1.8. We select MySQL 5.7.25 as the supported database to store the off-chain data.

We deployed the implemented prototype on five Alibaba Cloud[1] virtual machines (2vCPUs, 8-GB RAM) as client/central servers (one cloud server used as the central server and four cloud servers used as client servers) and four Raspberry Pi 3 boards configured to Raspbian 2018-11-3 as four client devices. Each client server is allocated to one hotel where installed one large air-conditioner (as client device). The data collected by the sensors in each air-conditioner include 70 parameters. After reducing the noisy data and removing the redundant parameters (from two compressors), 18 parameters are kept for machine learning as client data, which are listed as follows.

1) Evaporator inlet water temperature.
2) Evaporator outlet water temperature.
3) Condenser inlet water temperature.
4) Condenser outlet water temperature.
5) Evaporator cooling capacity.
6) Compressor inlet air temperature.
7) Compressor outlet air temperature.
8) Evaporator inlet air pressure.
9) Condenser outlet air pressure.
10) Exhaust air overheat temperature.
11) Main circuit's coolant level.
12) Opening size of the main coolant pipe valve.
13) Compressor's load.
14) Compressor's current.
15) Compressor's rotational speed.
16) Compressor's voltage.
17) Compressor's power.
18) Compressor's inverter's temperature.

For the experiments, each data record consists of 18 features as mentioned above and one label that indicates whether any fault occurred. A client server has one classifier, which is a four-layer neural network (NN). The first layer consists of 18 units corresponding to the 18 features. Then, there are two hidden layers that both have 150 units. The output layer is a *Softmax* function to classify normal data and abnormal data. Note that the hidden layers apply *ReLu* function as the activation function. The learning rate is set to 0.005.

C. Feasibility

To evaluate the feasibility of the proposed approach, we examined the implemented proof-of-concept of proposed architecture against the functional and nonfunctional requirements identified in Section III. Fig. 3 illustrates the workflow of federated learning and blockchian anchoring.

For the functional requirements: 1) the platform prototype is able to train local models on the client server hosted by each hotel and aggregate the local model updates into a global model for failure detection; 2) the operators of the hotels and the air-conditioner platform are able to verify the integrity of client data for a certain anchoring period by comparing with the Merkle tree root stored on blockchain; and 3) in addition,

[1]https://www.aliyun.com/

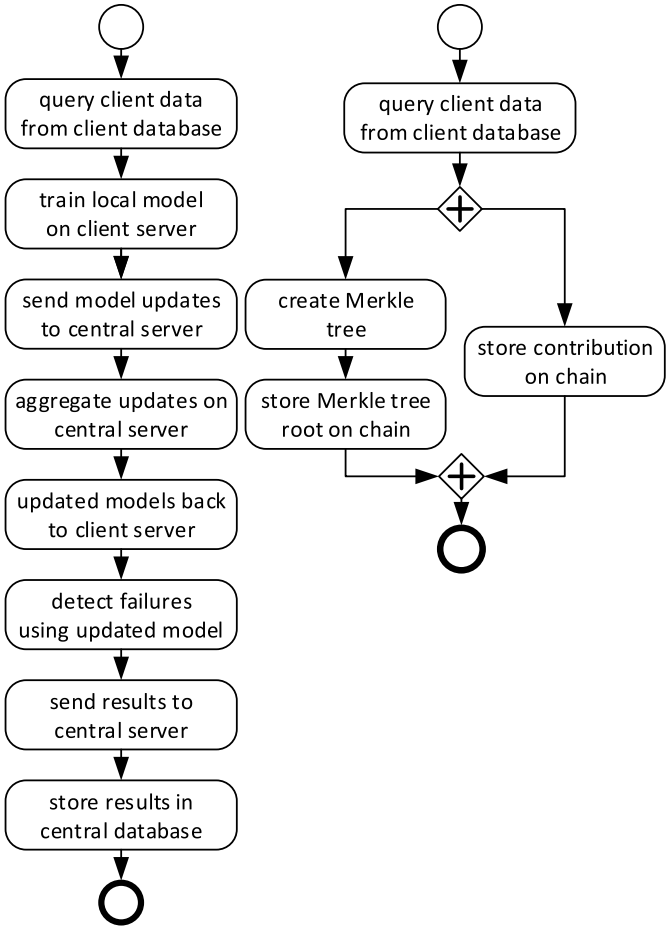

Fig. 3. Federated learning process and blockchain anchoring process.

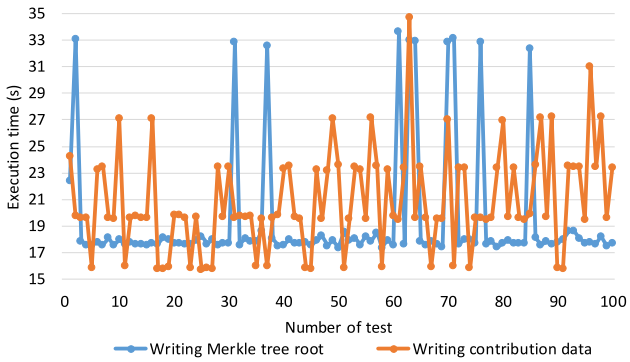

Fig. 4. Blockchain execution time.

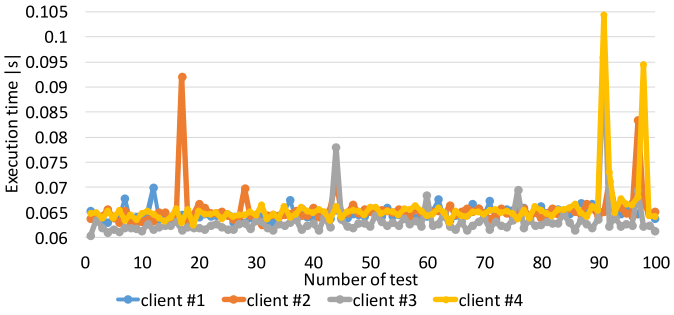

Fig. 5. Execution time of incentive mechanism.

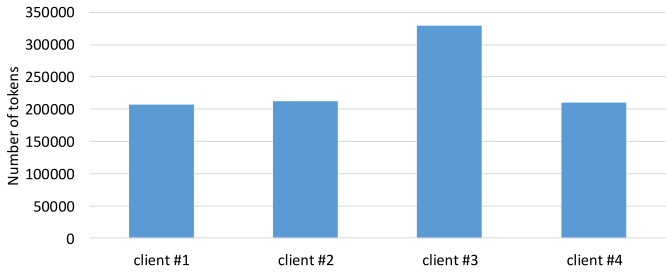

Fig. 6. Tokens given to each client.

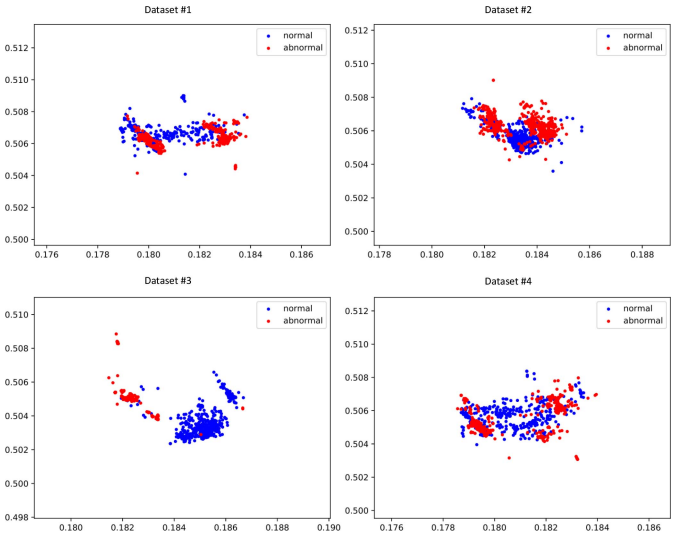

Fig. 7. Training data distribution after dimension reduction.

the hotels can be allocated with tokens as the reward for their contribution via *Incentive Registry SC*.

Regarding the nonfunctional requirements: 1) the air-conditioner data privacy is preserved by the federated learning setting, in which the central server only needs air-conditioner model updates rather than all air-conditioner usage information; 2) in particular, the applied federated learning is more advantageous than centralized model training on the bandwidth cost, if the size of training data is large; 3) the detection accuracy is maintained by a shared model, which is formulated by aggregating updates from models trained on each hotel (client server); and 4) air-conditioner data integrity is achieved via the anchoring protocol. When data auditing is needed, the data stored locally in each hotel can be again constructed to the Merkle tree structure and verified by comparing it with the root stored on the blockchain. Each hotel and the air-conditioner manufacturer maintain a full node to keep the blockchain infrastructure available, in the case that some nodes maybe out of service.

D. Quantitative Analysis—Accuracy and Performance

In the quantitative analysis, each client server stores about 1000 training records and 1000 testing records. The training records are distributed on client servers, while a global model is learned on the central server by aggregating local model updates. We first tested accuracy of the global model of failure detection on each individual client server with their own

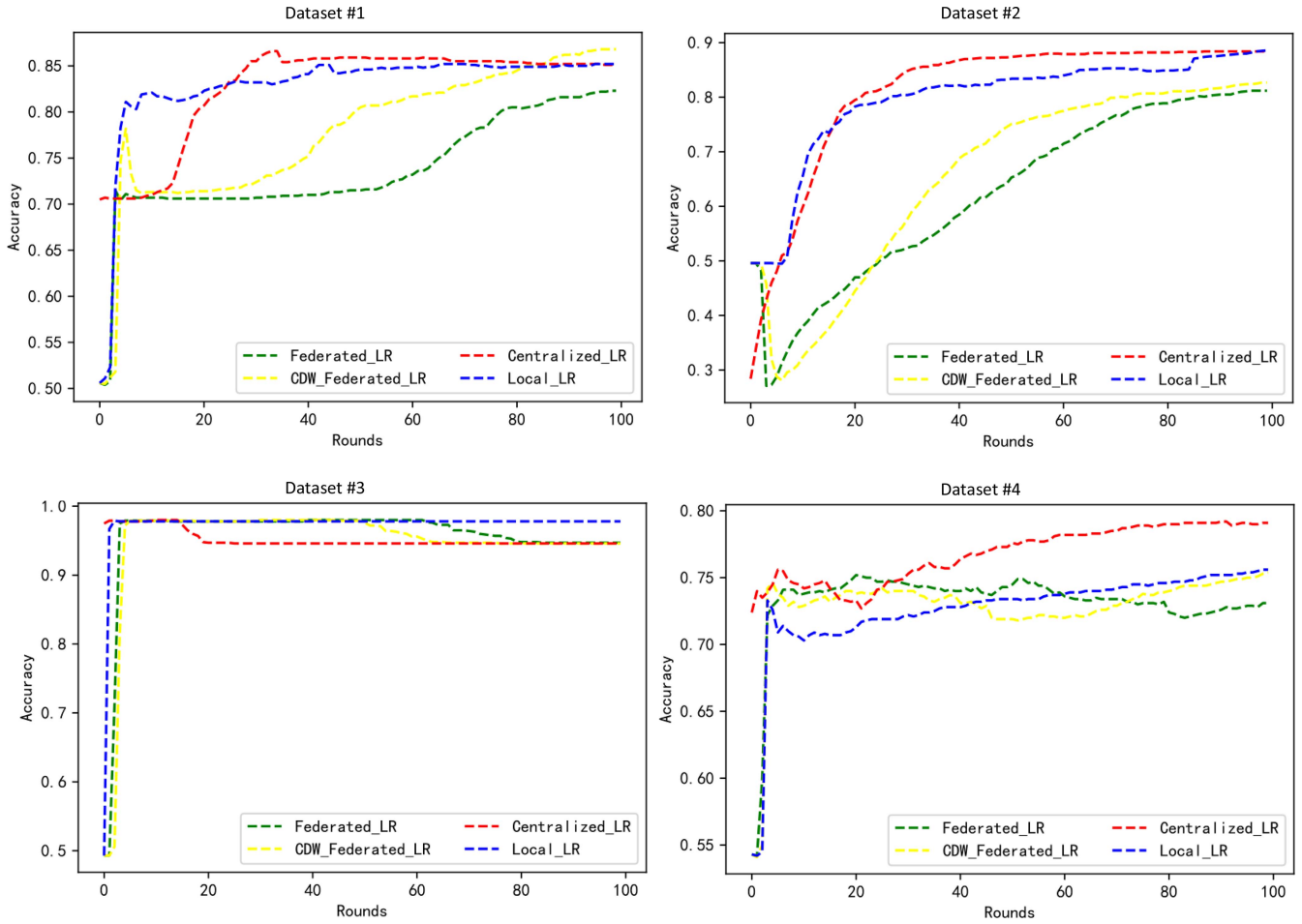

Fig. 8. Accuracy of LR model.

testing records. Moreover, we compared the accuracy of the global model trained using federated learning with the respective model trained using the traditional centralized learning and local learning. We tested two classification models, including logistic regression (LR) and NNs.

Test 1 (Measuring Blockchain Anchoring Time): The executing time of the anchoring protocol includes querying data from the database, creating a Merkle tree, and writing the Merkle tree root and contribution data on the blockchain. Fig. 4 only illustrates the execution time for writing Merkle tree root and contribution data to blockchain, since it costs most of the execution time for the anchoring protocol. We measured the transaction inclusion time which is for the transaction to be included in a block. Writing the Merkle tree root takes around 18 s, while the average time for writing contribution data is around 21 s. The writing latency on the blockchain is much longer than in the conventional database as it takes time to propagate transactions/blocks and achieve consensus. Since blockchain is used for auditing and incentives provision, the writing latency of blockchain is not a concern in this work. We have also examined the Merkle tree creation time, which takes about 250 ms. As the Merkle tree creation time is much less than transaction inclusion time, it is not included in Fig. 4.

Test 2 (Measuring Incentive Mechanism): The executing time of the incentive mechanism includes the time spent on calculating tokens rewarded and writing them on blockchain. We measured the executing time of the incentive mechanism. As shown in Fig. 5, the average execution time is around 0.065 s for all the four clients, which is efficient. Fig. 6 presents the tokens rewarded to each of the four clients with different data distance. Client #3 gains the largest number of tokens since it has the largest data distance. The results show that the incentive mechanism encourages clients to provide high quality training data.

Test 3 (Measuring Model Detection Accuracy): Before we measured the model prediction, we first examined the distribution of each client data set to check their distribution difference. Fig. 7 shows the distribution of positive class and negative class in each client data set. We observed that there is a distance from positive class to negative class in client #3, while other clients show a different pattern.

We measured the accuracy of LR and NNs, using proposed centroid distance weighted (CDW) federated learning method with CDW_FedAvg, standard federated learning with FedAvg, centralized learning and local learning, respectively, for 100 training rounds. For each round of federated learning, the model is trained locally on each client through 40 epochs and the model updates are sent to central server to produce a global model. Each client uses the global model to make failure prediction. For each round of centralized learning, a

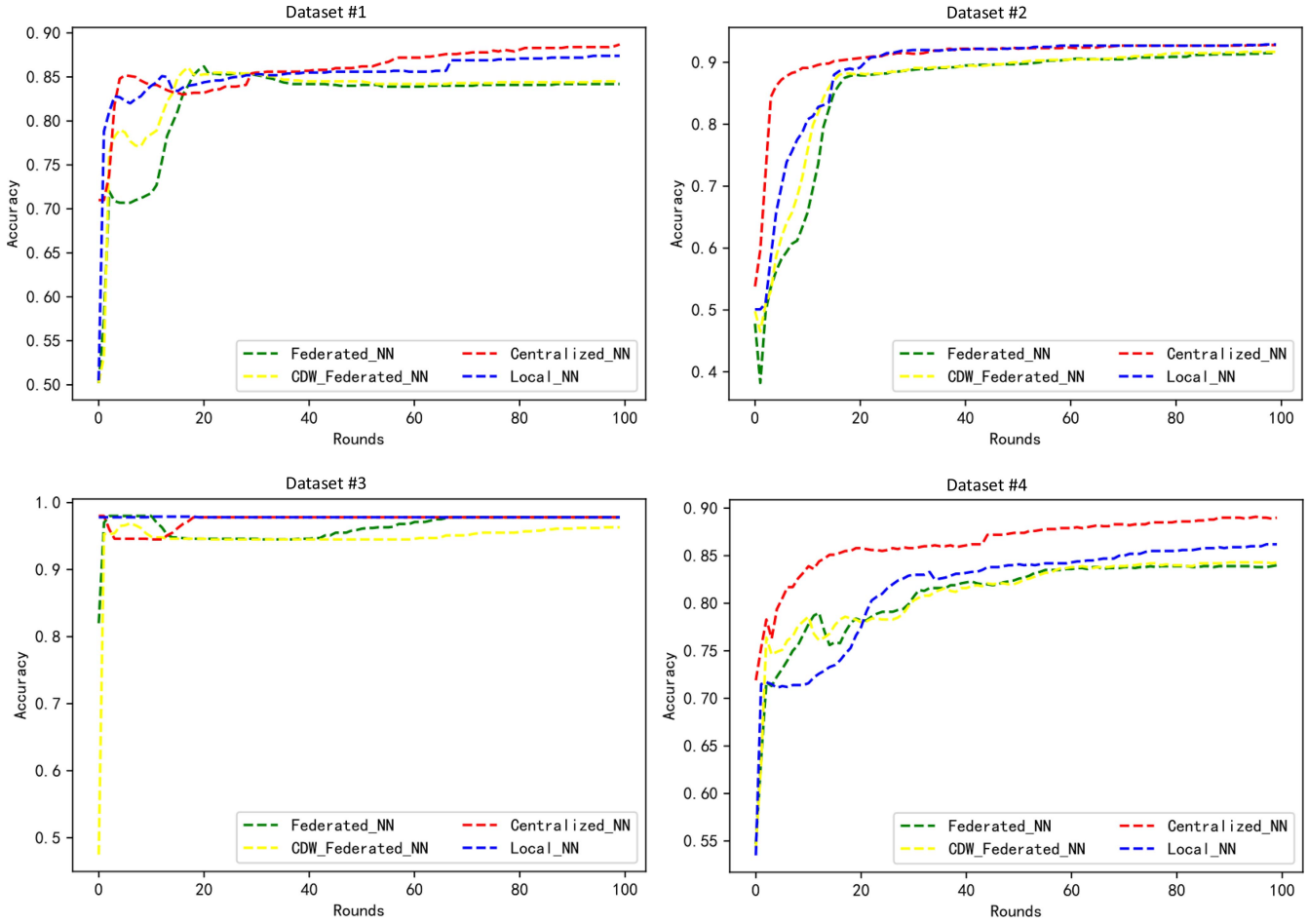

Fig. 9. Accuracy of NN model.

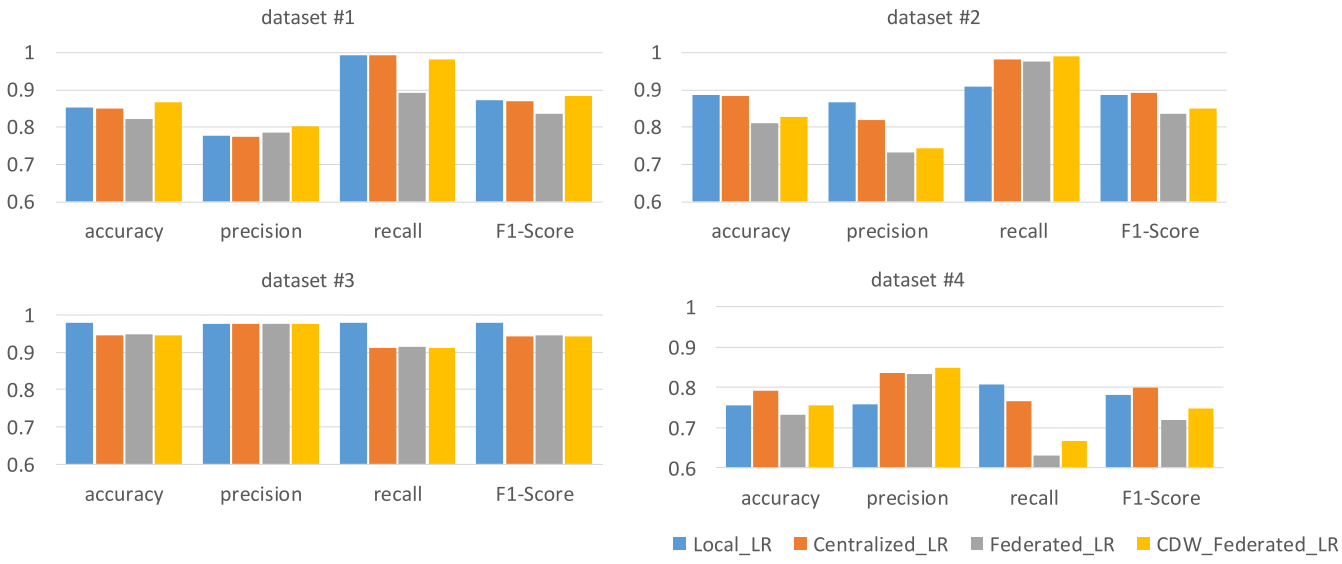

Fig. 10. Precision, recall, and F1-score of LR model.

central server trains the model through 40 epochs and each client uses this model to predict failures. Compared with the federated learning model that is trained on local training data set, the centralized training model is trained on an aggregated training data set that consist of local data sets from all clients. We tested the centralized model and federated learning model using the same test data set. For each round of local learning, each client trains a local model using local data set through 40 epochs and uses the local model to make detection.

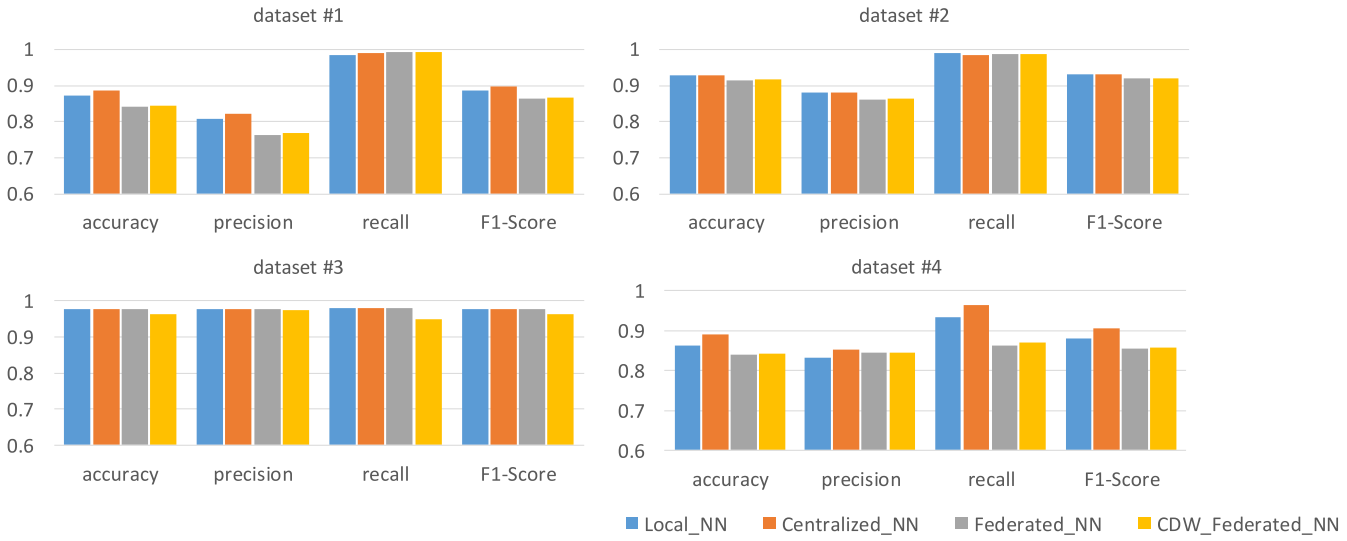

Fig. 11. Precision, recall, and F1-score of NN model.

The measurement results are shown in Figs. 8 and 9. The *x*-axis represents the number of training round. The *y*-axis means the percentage of correct detection (both true positives and true negatives). In Figs. 8 and 9, each subgraph includes four lines, which shows the accuracy of model using different learning methods and tested on the corresponding client data sets. For example, in Fig. 8 *Dataset #1*, *CDW_Federated_LR* represents the accuracy of the LR model, which was trained using the proposed CDW federated learning method with CDW_FedAvg and tested on client #1 data set. Overall, Figs. 8 and 9 show that the proposed CDW federated learning and standard federated learning can achieve satisfactory accuracy as local learning and centralized learning in most of the times. The accuracy of the models tested on clients differs from each other because the data distribution of the respective clients are different. Specifically, for both LR and NN, the accuracy of both the proposed CDW federated learning and standard federated learning tested on client #3 is the highest, while the accuracy of federated learning performed on client #4 is the lowest. Furthermore, we observe that NN performs better than LR in all types of learning. In addition, the proposed CDW federated learning performs better than the standard federated learning for LR, while the two federated learning algorithms perform similar for NN.

In addition to accuracy, we also measured three other metrics in the experiments, including precision, recall, and F1-score. Precision means the proportion of truly classified positive instances to the total classified positive instances, while recall means the ratio of truly classified positive instances to the total number of practical positive instances. F1-score is the harmonic mean of precision and recall. The formal definitions of precision, recall, and F1-score are shown in formula (2)–(4), respectively. We measured these three metrics of LR and NN using both federated learning and centralized learning. The results are illustrated in Figs. 10 and 11. The *x*-axis includes four metrics, while the *y*-axis is the value of three measured metrics. The results show that the proposed CDW federated learning method and the standard federated learning method performed best on the data set of client #3, which is similar to the results of accuracy tests. Compared with the standard federated learning method, the proposed CDW federated learning method shows an improvement in Fig. 10, while the improvement is slight in Fig. 11

TABLE I
COMMUNICATION OVERHEAD

Data Size	10^3	10^6	10^9
Centralized_LR	10^3	10^6	10^9
Fed_LR(1)	28800	28800	28800
Fed_LR(2)	57600	57600	57600
Fed_LR(3)	86400	86400	86400
Fed_LR(4)	115200	115200	115200

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{2}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{3}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{4}$$

$$F1 - \text{Score} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{5}$$

TP, TN, FP, and *FN* are the abbreviations of *True Positive, True Negative, False Positive,* and *False Negative*, respectively.

Test 4 (Measuring the Communication Overhead): We measured the communication overhead of LR. The results are shown in Table I, where Fed_LR(N) means *N* clients are chosen randomly to train the model in each round. In our implementation, the size of the LR model is 144 B and the number of training rounds is set to 100. For instance, in Fed_LR(3), the total communication overhead is $144 \times 3 \times 100 \times 2 = 86400$ B, where *2 means the models are gathered from every chosen client and then the global model is sent back to the clients after aggregating the client model updates. The table shows the overhead of centralized learning is much more than that of federated learning, because all data need to be sent to the

central server from the client servers in centralized learning, while only model updates are sent to the central server in federated learning. Therefore, the communication overhead of the federated learning is irrelevant to the size of the data. When the size of data set is extremely large, the communication overhead can be reduced significantly.

V. Conclusion

This article presented a blockchain-based federated learning approach for IIoT device failure detection. In the design, a central server coordinates client servers to train a shared global model for detection, with raw data stored locally. The architecture adopts blockchain to enable verifiable integrity of client data and incentives for client contribution. To reduce the impact of data heterogeneity, a novel CDW_FedAvg algorithm is proposed based on the distance between positive class and negative class of each client data set.

A proof-of-concept prototype is implemented using the proposed architecture in a real-world use case. We evaluate the approach in terms of feasibility, detection accuracy, and performance. The evaluation results show that the proposed approach achieve all the proposed objectives.

Although most of the federated learning modules are placed on client servers in this article, we plan to adapt the design by moving all the modules from client servers to client devices with more powerful computing and storage capabilities in our future work. Also, we plan to explore how to increase the trustworthiness of client devices for the aggregation process in further study.

References

[1] M. Banerjee, C. Borges, K. R. Choo, J. Lee, and C. Nicopoulos, "A hardware-assisted heartbeat mechanism for fault identification in large-scale IoT systems," *IEEE Trans. Depend. Secure Comput.*, early access, Jul. 16, 2020, doi: 10.1109/TDSC.2020.3009212.

[2] S. Seshadri *et al.*, "IoTCop: A blockchain-based monitoring framework for detection and isolation of malicious devices in Internet-of-Things systems," *IEEE Internet Things J.*, early access, Sep. 7, 2020, doi: 10.1109/JIOT.2020.3022033.

[3] J. Konecný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon. (2016). *Federated Learning: Strategies for Improving Communication Efficiency*. [Online]. Available: http://arxiv.org/abs/1610.05492

[4] S. K. Lo, Q. Lu, C. Wang, H. Paik, and L. Zhu. "A systematic literature review on federated machine learning: From a software engineering perspective," 2020. [Online]. Available: arxiv.abs/2007.11354.

[5] H. Kim, J. Park, M. Bennis, and S. Kim, "Blockchained on-device federated learning," *IEEE Commun. Lett.*, vol. 24, no. 6, pp. 1279–1283, Jun. 2020.

[6] J. Weng, J. Weng, J. Zhang, M. Li, Y. Zhang, and W. Luo, "DeepChain: Auditable and privacy-preserving deep learning with blockchain-based incentive," *IEEE Trans. Depend. Secure Comput.*, early access, Nov. 8, 2019, doi: 10.1109/TDSC.2019.2952332.

[7] U. Majeed and C. S. Hong, "FLchain: Federated learning via mec-enabled blockchain network," in *Proc. 20th Asia–Pac. Netw. Oper. Manag. Symp. (APNOMS)*, Sep. 2019, pp. 1–4.

[8] W. Shi and S. Dustdar, "The promise of edge computing," *Computer*, vol. 49, no. 5, pp. 78–81, 2016.

[9] H. B. McMahan, E. Moore, D. Ramage, and B. A. y Arcas. (2016). *Federated Learning of Deep Networks Using Model Averaging*. [Online]. Available: http://arxiv.org/abs/1602.05629

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

[11] Y. Zhao, J. Zhao, L. Jiang, R. Tan, and D. Niyato, "Mobile edge computing, blockchain and reputation-based crowdsourcing IoT federated learning: A secure, decentralized and privacy-preserving system," 2019. [Online]. Available: arxiv.abs/1906.10893.

[12] T. Yang *et al.* (2018). *Applied Federated Learning: Improving Google Keyboard Query Suggestions*. [Online]. Available: http://arxiv.org/abs/1812.02903

[13] M. Chen, R. Mathews, T. Ouyang, and F. Beaufays, "Federated learning of out-of-vocabulary words," 2019. [Online]. Available: arxiv.abs/1903.10635.

[14] S. Ramaswamy, R. Mathews, K. Rao, and F. Beaufays, "Federated learning for emoji prediction in a mobile keyboard," 2019. [Online]. Available: arxiv.abs/1906.04329.

[15] M. J. Sheller, G. A. Reina, B. Edwards, J. Martin, and S. Bakas, "Multi-institutional deep learning modeling without sharing patient data: A feasibility study on brain tumor segmentation," in *Proc. Int. MICCAI Brainlesion Workshop*, 2019, pp. 92–104.

[16] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2019, pp. 1–7.

[17] Y. Han and X. Zhang, "Robust federated learning via collaborative machine teaching," in *Proc. 34th AAAI Conf. Artif. Intell.*, vol. 34, 2020, pp. 1–8.

[18] J. Pang, Y. Huang, Z. Xie, Q. Han, and Z. Cai, "Realizing the heterogeneity: A self-organized federated learning framework for IoT," *IEEE Internet Things J.*, early access, Jul. 7, 2020, doi: 10.1109/JIOT.2020.3007662.

[19] S. Nakamoto. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*. Accessed: Oct. 5, 2020. [Online]. Available: https://bitcoinsv.io/bitcoin

[20] Y. Jiang, Y. Zhong, and X. Ge, "Smart contract-based data commodity transactions for industrial Internet of Things," *IEEE Access*, vol. 7, pp. 180856–180866, 2019.

[21] S. Zheng, T. Han, Y. Jiang, and X. Ge, "Smart contract-based spectrum sharing transactions for multi-operators wireless communication networks," *IEEE Access*, vol. 8, pp. 88547–88557, 2020.

[22] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, "Blockchain and federated learning for privacy-preserved data sharing in industrial IoT," *IEEE Trans. Ind. Informat.*, vol. 16, no. 6, pp. 4177–4186, Jun. 2020.

[23] J. Kang, Z. Xiong, D. Niyato, S. Xie, and J. Zhang, "Incentive mechanism for reliable federated learning: A joint optimization approach to combining reputation and contract theory," *IEEE Internet Things J.*, vol. 6, no. 6, pp. 10700–10714, Sep. 2019.

[24] X. Bao, C. Su, Y. Xiong, W. Huang, and Y. Hu, "FLchain: A blockchain for auditable federated learning with trust and incentive," in *Proc. 5th Int. Conf. Big Data Comput. Commun. (BIGCOM)*, Aug. 2019, pp. 151–159.

[25] G. Kotonya and I. Sommerville, *Requirements Engineering: Processes and Techniques*, 1st ed. New York, NY,USA: Wiley, 1998.

[26] S. Caldas *et al.*, "Leaf: A benchmark for federated settings," 2019. [Online]. Available: arxiv.abs/1812.01097.

[27] V. Buterin. (2013). *Ethereum White Paper: A Next Generation Smart Contract & Decentralized Application Platform*. Accessed: Oct. 5, 2020. [Online]. Available: https://ethereum.org/en/whitepaper/

Weishan Zhang (Member, IEEE) received the Ph.D. degree in mechanical manufacturing and automation from Northwestern Polytechnical University, Xi'an, China, in 2001.

He is a Full Professor and the Deputy Head for Research with the Department of Software Engineering, School of Computer and Communication Engineering, China University of Petroleum, Qingdao, China. His research interests include big data platforms, pervasive cloud computing, service-oriented computing, and federated learning.

Qinghua Lu (Senior Member, IEEE) received the Ph.D. degree from the University of New South Wales, Sydney, NSW, Australia, in 2013.

She is a Senior Research Scientist with Data61, CSIRO, Sydney, NSW, Australia. Before she joined Data61, she was an Associate Professor with the China University of Petroleum, Qingdao, China. She formerly worked as a Researcher with National ICT Australia, Sydney. She has published over 100 academic papers in international journals and conferences. Her recent research interest includes architecture design of blockchain systems, blockchain for federated/edge learning, self-sovereign identity, and software engineering for machine learning.

Qiuyu Yu received the B.Sc. degree from China University of Petroleum (East China), Qingdao, China, in 2017, where she is currently pursuing the postgraduation degree with the College of Computer Science and Technology.

Her research interests include the software architecture of blockchain and federated learning.

Zhaotong Li received the B.Sc. degree from China University of Petroleum (East China), Qingdao, China, in 2018, where he is currently pursuing the postgraduation degree with the College of Computer Science and Technology.

His research interests include data mining and federated learning.

Yue Liu received the M.S. degree from China University of Petroleum (East China), Qingdao, China, in 2020. He is currently pursuing the Ph.D. degree with the University of New South Wales, Sydney, NSW, Australia, and CSIRO, Sydney.

His research interests include blockchain as a service, blockchain governance, and self-sovereign identity.

Sin Kit Lo received the M.S. degree from the University of Malaya, Kuala Lumpur, Malaysia, in 2020. He is currently pursuing the Ph.D. degree with the University of New South Wales, Sydney, NSW, Australia, and CSIRO, Sydney.

His research interests include federated learning, blockchain, and software architecture.

Shiping Chen (Senior Member, IEEE) received the Ph.D. degree from the University of New South Wales, Sydney, NSW, Australia, in 2001.

He is a Principal Research Scientist with Data61, CSIRO, Sydney. He also holds an Adjunct Associate Professor with the University of Sydney, Sydney, through teaching and supervising Ph.D./Master students. He has been working on distributed systems for over 20 years with focus on performance and security. He has published over 100 research papers in these research areas. He is actively involved in computing research community through publications, journal editorships, and conference PC services, including WWW, EDOC, ICSOC, and IEEE ICWS/SCC/CLOUD. His current research interests include secure data storage & sharing and secure multiparty collaboration.

Xiwei Xu (Member, IEEE) received the Ph.D. degree from the University of New South Wales (UNSW), Sydney, NSW, Australia, in 2012.

She is a Senior Research Scientist with Architecture& Analytics Platforms Team, Data61, CSIRO. She is also a Conjoint Lecturer with UNSW. She started working on blockchain since 2015. She is doing research on blockchain from software architecture perspective, for example, tradeoff analysis, and decision making and evaluation framework. Her main research interest is software architecture. She also does research in the areas of service computing, business process, and cloud computing and dependability.

Liming Zhu received the Ph.D. degree from the University of New South Wales, Sydney, NSW, Australia, in 2007.

He is a Research Director with Data61, CSIRO, Sydney. He is also a Conjoint Full Professor with the University of New South Wales, Sydney. His research program has more than 300 people innovating in the area of big data platforms, computational science, blockchain, regulation technology, and privacy and cybersecurity. He has published more than 200 academic papers on software architecture, secure systems and data analytics infrastructure, and blockchain.

Dr. Zhu is the Chairperson of Standards Australia's blockchain and Distributed Ledger Committee.