IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 4 September 2023, accepted 23 September 2023, date of publication 27 September 2023,
date of current version 5 October 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3320045

RESEARCH ARTICLE

FlwrBC: Incentive Mechanism Design for Federated Learning by Using Blockchain

NGUYEN TAN CAM AND VU TUAN KIET
University of Information Technology, Ho Chi Minh City, Vietnam
Vietnam National University, Ho Chi Minh City, Vietnam

Corresponding author: Nguyen Tan Cam (camnt@uit.edu.vn)

This research was supported by The VNUHCM-University of Information Technology's Scientific Research Support Fund.

ABSTRACT The growth of information technology has resulted in a massive escalation of data and the demand for data exploration, particularly in the machine learning sector. However, machine learning raises concerns about data privacy because algorithms require large amounts of data to learn and make accurate predictions. Such data often contain personal information about individuals, and there is a risk that this information could be accessed or misused by unauthorized parties. It is crucial for organizations that use machine learning to prioritize personal data protection and ensure that appropriate safeguards are in place to prevent privacy breaches. Federated learning (FL) and blockchain technology are two increasingly popular approaches to distributed computing. Federated learning is a distributed machine-learning approach that trains machine-learning models on decentralized datasets without centralizing the data. Federated learning offers several benefits, including improved data privacy. Ensuring the benefit of clients in federated learning is vital for the success of this distributed machine learning approach, especially when combined with blockchain technology, as it offers a secure and transparent way to store and verify data. In this study, we propose a combination of federated learning and blockchain as a solution to some of the challenges faced by both approaches. By leveraging the decentralized nature of federated learning and the security and transparency of blockchain, our approach tends to overcome issues such as data privacy and trustworthiness of results. The evaluation results demonstrated that the proposed approach has many potential applications in various domains.

INDEX TERMS Artificial intelligence, blockchain, federated learning, machine learning, incentive mechanism.

I. INTRODUCTION

With the rise of information technology, the world is generating data at an unprecedented rate, leading to an increased demand for data exploration and machine learning. However, this demand is accompanied by growing concerns regarding data protection and privacy, which hinders the use of traditional centralized machine learning methods. Traditional methods involve collecting and storing data on a central server, where machine learning algorithms are executed to produce a model after training. In this approach, user data protection and privacy depend entirely on the server management. Moreover, traditional methods may require assistance in data transmission and storage. Traditional techniques can no longer handle massive datasets because the data size is increasing. Consequently, cloud and edge computing have become valuable tools with powerful and flexible computational capabilities. Cloud or edge computing technologies provide infrastructure with thousands of computing servers and offer multimachine computation capabilities worldwide [1]. For example, a single vehicle can produce hundreds of gigabytes of data via its sensors in the autonomous vehicle industry. The transfer and storage of data from all vehicles on a single server are expensive and impractical [2]. Moreover, data generated from different sources are often decentralized, making it difficult to collect and store

The associate editor coordinating the review of this manuscript and approving it for publication was Sedat Akleylek.

© 2023 The Authors. This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

data on a single machine. This is particularly true when dealing with big data on multimachine computing servers. A single machine does not have sufficient storage capacity for large amounts of data, and transferring data between devices can result in substantial network overhead and congestion. Additionally, data samples cannot be spread across networks for information and personal privacy. Consequently, there is a need for a suitable computational framework and a distributed method for big data.

In 2017, Google introduced a federated learning technique to address this issue. In this decentralized approach, clients in a learning network communicate only the parameters of the training model with the server, thereby eliminating the need for data transmission [3]. Federated Learning aids in reducing the amount of data in communication, and mitigates threats related to data privacy and protection. Centralized storage strategies can lead to privacy risks. A federated learning framework is used to protect privacy. This training was performed using local model training on the end-user's device. The server computes only global parameters, leading to effective privacy preservation [4].

However, another issue has been raised: the exploitation of user data collected by organizations and businesses for analysis and machine learning purposes and the integrity of the learning process might not be guaranteed [5]. These data are generally exploited for free, without users receiving any economic benefits, even though organizations and companies are applying traditional machine learning or federated learning methods. Furthermore, the previous schemes ignored the evaluation of the performances contributed by specific client datasets. Another aspect of fair trading of user data in traditional transaction schemes is the presence of a centralized third party (such as banks and financial companies), which poses a threat of disputation, dubiety, privacy, and so on. Scheme-based blockchain technology is a fair and reliable solution for this issue [6].

The combination of blockchain technology and federated learning can help prevent personal data leakage and provide users with a corresponding profit for their data contribution to the machine learning of organizations and businesses. Furthermore, this combination reduces the risk of a Single Point of Failure, because the system is less dependent on a central server. Based on this discussion, this paper studies the implementation of federated learning, focusing on aggregating the weights contributed by clients in the network. In this study, we applied smart contracts to a simulated blockchain to record and ensure the benefits to client nodes during federated learning.

This study has the main contribution as follows:

- Propose a machine-learning framework that applies federated learning to preclude data leakage and utilizes blockchain to award data contributors.
- Test scenarios were conducted to evaluate and compare the effectiveness of the proposed framework and traditional machine learning using the CIFAR-10 dataset.

The remainder of this paper is organized as follows. An introduction is presented in section of this paper. Section II presents the related work and background. Section III describes the proposed system. The evaluations are presented in Section IV. Finally, conclusions and future work are discussed in Section V.

II. RELATED WORKS

A. NETWORK STRUCTURE OF FEDERATED LEARNING

In federated learning, the two main network structures are the client-server and peer-to-peer modes [7]. In peer-to-peer mode, nodes in the network cluster can only exchange data with their neighbors. No master node handles information broadcast or transfer [8]. In the client-server mode, a master node controls the information exchange between all devices. The master node receives messages from worker nodes, processes all variables, and then sends or broadcasts information to specific or all nodes for small-problem optimization [9].

Many algorithms in distributed environments randomly divide datasets into small ones that are stored on different nodes. They processed the datasets and trained the local models in parallel. Nonetheless, this approach does not consider the differences between the data collected from different resources on a single machine. To address this issue, we suggest using a global constraint to ensure a distributed model regardless of whether the data distribution is the same.

In this study, we discuss a distributed classification method using a client-server mode in federated learning. In this approach, each slave node trains its local model. Subsequently, the model parameters are sent to the master node. The master node then receives all the local model parameters. Subsequently, the global model parameters are updated using a collaborative mechanism. The master node then broadcasts the updated global model parameters to other nodes. This process was repeated until the optimization problem reached a global consensus.

B. COMPARISON BETWEEN THE CENTRALIZED AND FEDERATED LEARNING

Asad et al. [10] indicated the differences in executions between centralized, distributed, and federated learning and measured the performance of those approaches in solving classification problems.

1) CENTRALIZED LEARNING

This method involves connecting the participants to a central server and uploading their data. The server then uses machine learning algorithms to train a model based on these data. The advantage of this method is that there is no requirement for participants' resources during training, because the server performs the task entirely. However, this method also risks client data, because the server can be attacked or leak information. Additionally, the high volume of data involved in

the transmission process can result in the probability of data transmission failure.

2) DISTRIBUTED LEARNING

Distributed learning (distributed machine learning) is designed to solve complex problems with large datasets, making it more efficient and scalable than centralized learning. Edge computing has recently gained popularity owing to its ability to process applications, including basic machine-learning algorithms. This approach can be used to build public cloud-computing services for big data applications [11]. Distributed machine learning also implements algorithms such as traditional machine learning, but these tasks are performed independently and separately on each participant's machine. Initially, the server provides pre-trained models to clients. After the clients use their own datasets to train the models, the participating devices send the model parameters, w^i to the server. After a predetermined number of parameter exchanges, the server calculates the global model parameter w and performs testing, and updates the global model. Clients cannot use the results from other clients before receiving the global model [12].

3) FEDERATED LEARNING

Machine learning in federated learning (FL) operates similarly to distributed learning, with some studies viewing federated learning as a form of distributed machine learning [13] [14], [15], [16], [17], [18], [19]. However, in federated learning, the client runs training independently from other clients in the network. Each client declares local epochs when participating in the model training. After the local epochs are completed, the local model sends its model parameters, w^i. After receiving the local model parameters from all participating clients in that round, the server performs computations to update the global parameters, and sends the updated global model parameters back to all clients for the next round. In federated learning, this process continues until the global model reaches the desired accuracy or a specified number of rounds is reached.

C. FEDERATED LEARNING WITH BLOCKCHAIN

In traditional approaches, a centralized third party is used for data exchange services to negotiate the settlement of trade between data providers and consumers. This approach suffers from issues such as extra settlement fees, a single point of failure, and ambiguity in settlement details. To overcome these limitations, blockchain technology has been proposed as a new settlement model [17], [18], [20], [21], [22]. Blockchain smart contracts can be adopted to build a fair and autonomous settlement model. It also provides and proposes an optimized fault-tolerant consensus protocol. This approach can be used to achieve an identical shared ledger that records all transactions.

Kim indicated that the effectiveness of data privacy protection in the FL model depends on the server. If a server contains malware, it can affect the accuracy of the global model. It can potentially attack client data during local model parameter exchange [23]. Additionally, Feng Yu and Hui Lin pointed out that clients in a federated learning network have different configurations and datasets in practice, leading to various contributions to the global model. Hence, if these clients do not participate in federated learning, they will negatively affect the accuracy of the global model. Therefore, creating a mechanism that encourages clients to actively participate in federated learning is appropriate and positively affects the system [24].

Feng Yu and Hui Lin also suggested that during machine learning operations, some clients may accidentally or deliberately insert malware into the system, resulting in failure of the entire system [24]. By taking advantage of local model parameter exchange with the server, malware can be uploaded and spread within the system. Therefore, updates from the local model must be recorded for reference in case of an attack. Another scenario that leads to unexpected system failure could be the instability of communication between the server and clients.

Regarding incentive mechanisms, smart contracts seem to be a powerful tool for executing contracts without relying on a trusted intermediary. However, a disadvantage is that all transactions related to a smart contract are visible to all nodes in the blockchain, potentially raising privacy concerns [25]. To solve this issue, Zether [26] introduced a decentralized payment system that utilizes privacy-preserving smart contracts in an account-based model. This innovative system enables the concealment of account balances and the use of confidential transactions to update the associated balances, while keeping both transaction values and balances hidden. Zether is fully compatible with Ethereum and similar smart contract platforms.

In conclusion, the current client-server machine learning model has three challenges to address: (1) the system architecture heavily depends on the server, leading to Single Failure Point issues; the whole system will collapse if the server is attacked or malfunctions; (2) the reliability of participating clients and untrustworthy clients may attack and exploit the server's data or other peer clients during the exchange of model parameters; (3) although FL does not require sharing personal data, the entire system benefits from it, and businesses seek to profit from the model's accuracy; therefore, sharing the benefits to encourage clients to participate in training the system is appropriate and necessary.

The BlockFL (see Figure 1) model introduced by H. Kim and Jihong Park is summarized as follows: (1) clients calculate and update their local models to the miners in the main blockchain; (2) the miners are responsible for exchanging and verifying the posted local models and then running the consensus algorithm. (3) When the consensus algorithm was completed, a new block was created to verify the recorded local models. Finally, the newly created block contains the updated global model. The block is entered into

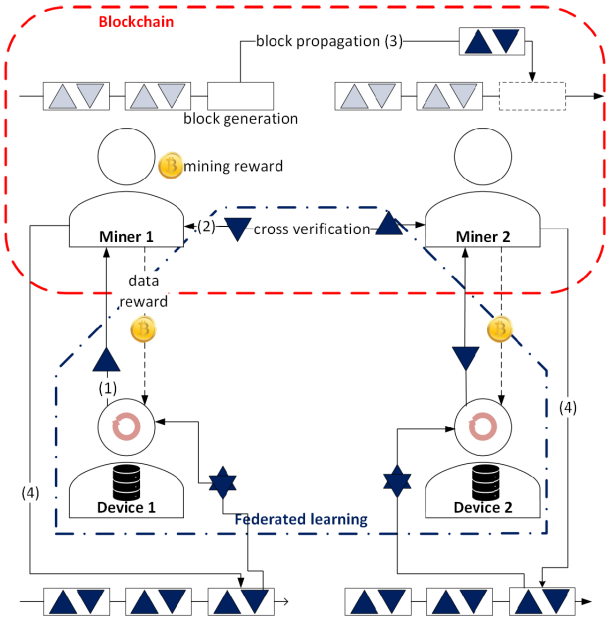

FIGURE 1. Description of BlockFL architecture.

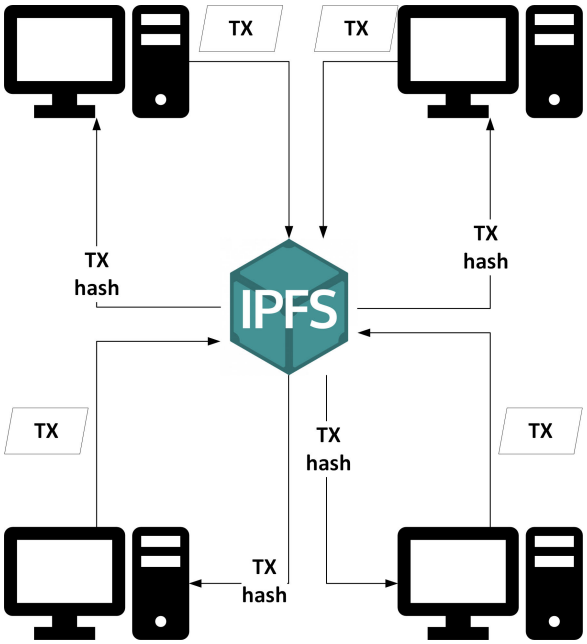

FIGURE 2. The IPFS-based data storage working model.

the distributed ledger, and the machines in the system use that model to calculate for the next round [23].

The above model results in the following experimental results: In terms of accuracy, BlockFL provides similar results to the traditional federated learning models; however, BlockFL takes longer to complete. The latency of BlockFL increases when the number of miners increases, because more time is spent on cross-verification and creating a new block. Regarding anti-malware, in BlockFL, a miner containing malware will only affect the update of the global model, whereas other normal miners in the system will recover.

D. DISTRIBUTED FILE STORAGE USING IPFS AND BLOCKCHAIN

Several essential applications use blockchain technology in distributed structures to ensure immutability, availability, and security. However, as the volume of transactional data increases, these applications are facing storage challenges. The number and size of transactions in a block on a blockchain are increasing owing to their immutable and append-only nature. Transactions growing in a block create the problem of storage and access for block transactions [27].

The Interplanetary File System (IPFS) is the foundation of Web 3.0, and it provides a decentralized file storage system and a technique for accessing stored files using content addressing. It connects to a P2P network and calculates a unique hash for each file that is accessible to all nodes in the network; the hash changes when the file is updated. The IPFS eliminates the need for centralized authorities and prevents the exploitation of personal information by a central entity. Privacy is enhanced using two different key pairs: public/private encryption keys provided by the Ethereum platform and a symmetric key for encrypting an IPFS profile when users choose to make some of their data private [28].

IPFS is a distributed data storage system that assigns a unique hash to each stored file using content addressing. It offers high throughput. Additionally, it offers an efficient storage model. In addition, it provides concurrent access. The hash created by the IPFS is 46 bytes long. The hash returned by the IPFS and the transaction data in the IPFS are stored in a blockchain block.

A data storage model using IPFS was proposed for blockchain to provide efficient storage and take advantage of IPFS features. The miners in this model collect transactions. Valid transactions were put into the mining pools. They are stored in a distributed network using the IPFS. During mining, the IPFS network provides a unique hash for the stored transactions. This method can be used to create new blocks. This transaction can be accessed using the hash value, which is known as content-addressable access. The proposed approach maintains the actual access name of the transaction (see Figure 2).

The IPFS is a distributed file system that uses content-addressed storage to store and retrieve files, which makes IPFS more efficient and secure than traditional file systems, as files are not stored on a single server and can be accessed from anywhere in the world. One of the key benefits of IPFS is its low latency. ($<$1ms) which is significantly lower than traditional file systems [29]. IPFS uses a distributed network of nodes to store files, so there is no single point of failure, making IPFS ideal for applications that require high availability and low latency, such as streaming media and gaming. Another key benefit of IPFS is its high reliability. IPFS files are stored redundantly across the network, so they

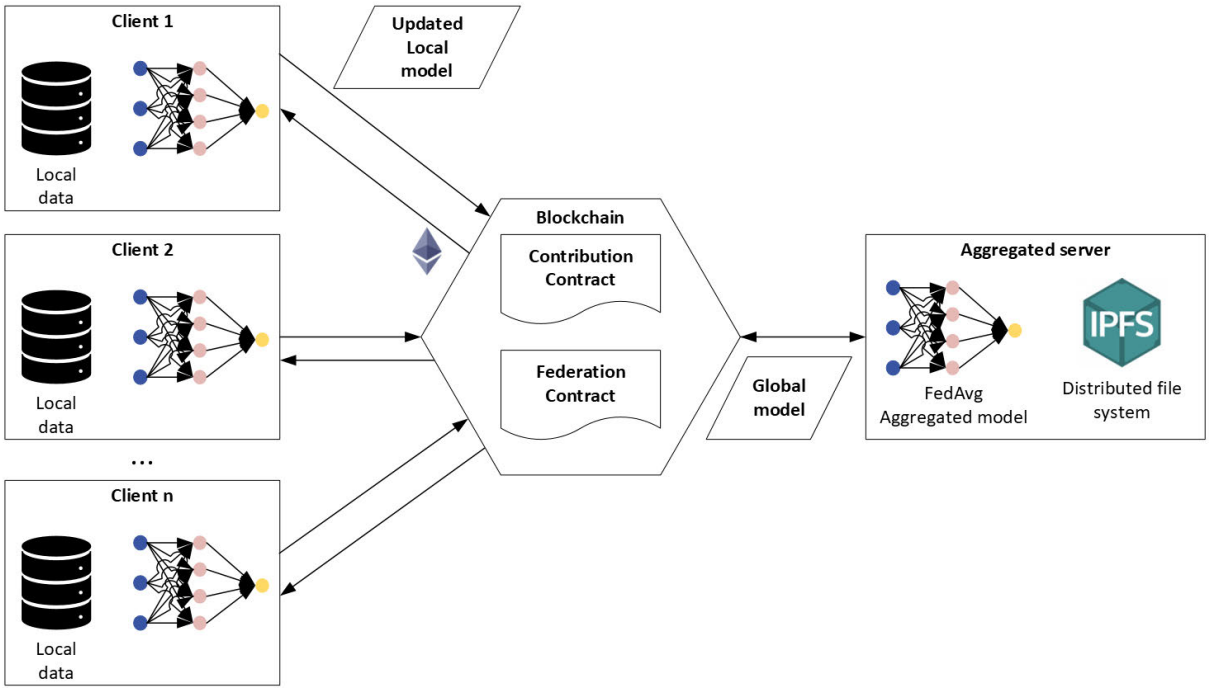

FIGURE 3. Description of FlwrBC architecture.

are highly resistant to corruption or loss [29]. IPFS is thus ideal for storing critical data, such as medical records and financial documents.

The IPFS can store and retrieve blockchain data, reducing the blockchain's storage cost while improving its security and reliability. The low latency and high reliability of IPFS are made possible by the underlying technology. IPFS uses a distributed hash table (DHT) to store files. DHT is a distributed database that maps file content to a unique identifier called a hash. This hash is used to locate the files in the network. The DHT is self-organizing, so it can dynamically adapt to changes in the network, meaning that the IPFS is scalable and fault-tolerant. IPFS also uses various techniques to improve performance, such as caching and replication, which helps ensure that files can be retrieved quickly and reliably.

III. PROPOSED SYSTEM

A. OVERALL FRAMEWORK

The framework developed in this study is based on Flower's federated learning framework [30] and is combined with the Ethereum blockchain named FlwrBC (see Figure 3). In previous studies, such as BlockFL [23] and BFL [24], the authors also pointed out that applying blockchain to peer-to-peer federated learning models incurs a significant gas fee and time when the number of machines participating is large. To address this issue, FlwrBC applies a blockchain to a client-server model. To solve the single-point failure problem of the model, the global model parameters are stored in a distributed storage system, and a cloud server is proposed for use as a server. An SSL layer was added between the client and server to ensure a secure connection. However, within the scope of this study, the SSL and cloud-server settings were simplified.

The FlwrBC is shown in Figure 3. The system has three main components: clients participating in federated learning, blockchain, and an aggregated server. Clients have separate datasets, and after training the model with their local datasets, they send the updated local model to the blockchain. Two smart contracts are implemented in the simulated blockchain: the Contribution Contract and the Federation Contract. With the Contribution Contract, after the client trains and updates the model parameters, the parameters (weights) are hashed and encoded before uploading them to the blockchain. Owing to the characteristics of blockchain, these weights ensure traceability and immutability. In addition, the system relies on the data size of clients to log and calculate rewards. On the side of the aggregated server, the server sends the global model parameters to the Federation Contract after receiving the weights from the clients and to perform global model aggregation. The Federation contract also performs hashing and stores the global model. In the design, the IPFS is used as a distributed file storage system to help the model withstand attacks if the storage system of the aggregated server is compromised.

The program flow is shown in Figure 4. The aggregated server and clients are authenticated in the blockchain, based on the address of the corresponding account. To start a session, (1) the aggregated server sends an API and (2) activates an event on the blockchain to notify clients within the blockchain. Upon receiving the notification, clients (3) activate the client program and (4) preprocess their

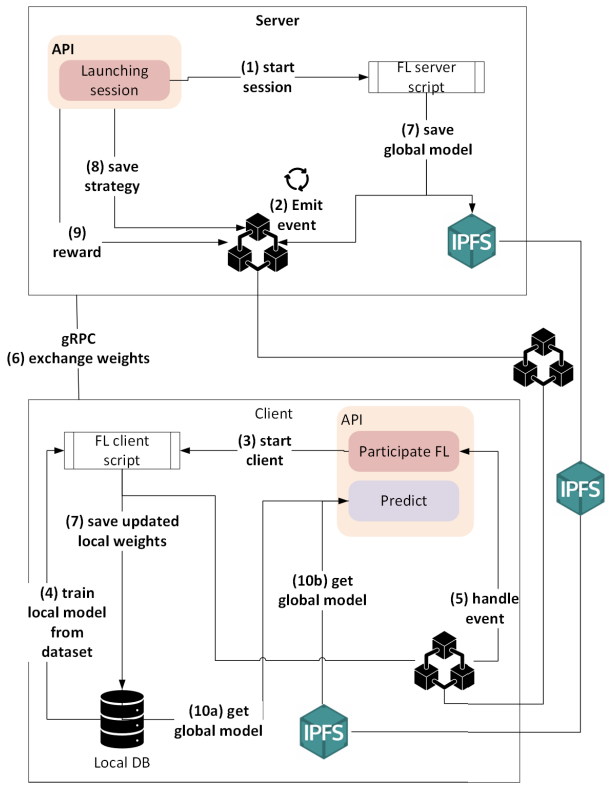

FIGURE 4. Description the program flow of FlwrBC.

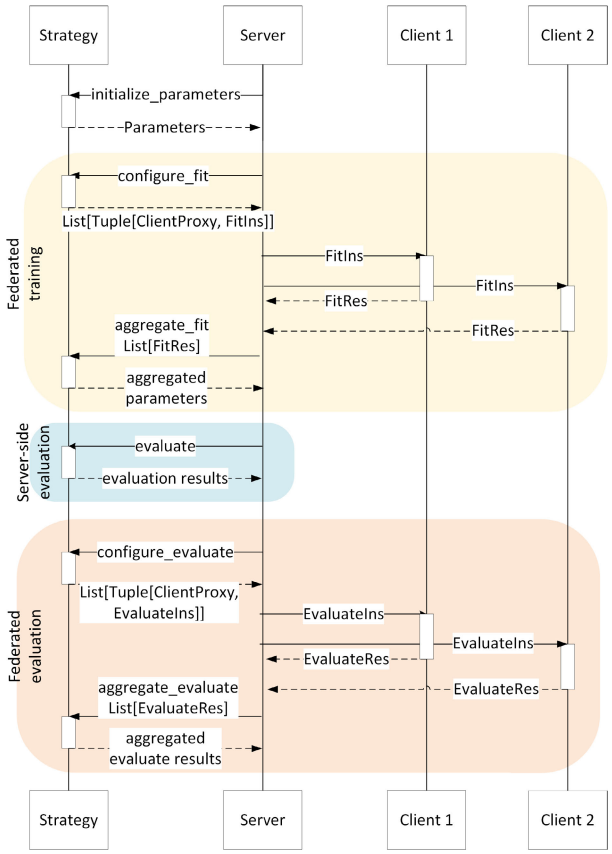

FIGURE 5. Data flow in federated learning mechanism of FlwrBC.

datasets. (6) At this point, there are two cases: if the server already has parameters from the previous session, it distributes them to all the clients. If the server does not have the model parameters from the previous session, it requests these parameters from a random client and distributes them to the remaining clients. In each federated learning session, (7) the server stores the global model parameter aggregation results and the corresponding hash chain on the IPFS distributed storage system, and (8) algorithmic information stored on the blockchain to facilitate future tracing. (7) After local training, the client saves the new local parameter information to the local database and then submits the parameters, associated hash chain, and related information to the blockchain through a Contribution Contract smart contract. (9) After the federated learning session was completed, each participant client was rewarded with a token amount corresponding to the data point contribution in that session. (10a) To use the results of federated learning, clients save the final global model parameters in their local database for prediction at the end of each session. (10b) If a new device wants to use the global for prediction but has yet to participate in the training, it can obtain the model from IPFS for its prediction.

B. THE MECHANISM OF FEDERATED LEARNING IN FLWRBC

Figure 5 shows the server-client communication process. This is divided into three sections: strategy, server, and client. A strategy is the federated learning algorithm that runs on the server, which decides how to sample clients, configure for training, aggregate updates, and evaluate the models. The *initialize_parameters* are called only once at the beginning of execution by the server. Consequently, the parameters are passed to *initial_parameters*if there are any. If no parameters are returned from *initialize_parameters,*the server arbitrarily asks one client in the network to provide its *parameters.*In the federated training stage, the server calls *configure_fit* to select clients and decides the instructions to send to these clients. The return value is a list of tuples in which each item represents an instruction (*FitIns*) sent to a specific client (*ClientProxy*). For each instruction sent to clients selected by *configure_fit*, the clients will result and return *FitRes* to the server. In this process, the failure is not promised to be carefree, and the server thus receives a list of successes and failures as *List[FitRes]*. The *aggregate_fit* method returns *aggregated_parameters*if *aggregated_fit*decides that the result is valid, or it can be concluded that the result is insufficient for aggregation if there are too many failures in the list. The framework has two approaches to evaluation: server-side evaluation and client-side evaluation. In client-side evaluations such as configure_fit, *configure_evaluate*decides how to sample those clients and sends instructions to the corresponding clients. *aggregate_evaluate* is then responsible for aggregating the list of successes and failures of evaluation resulting from clients. In server-side evaluations, having both evaluate and

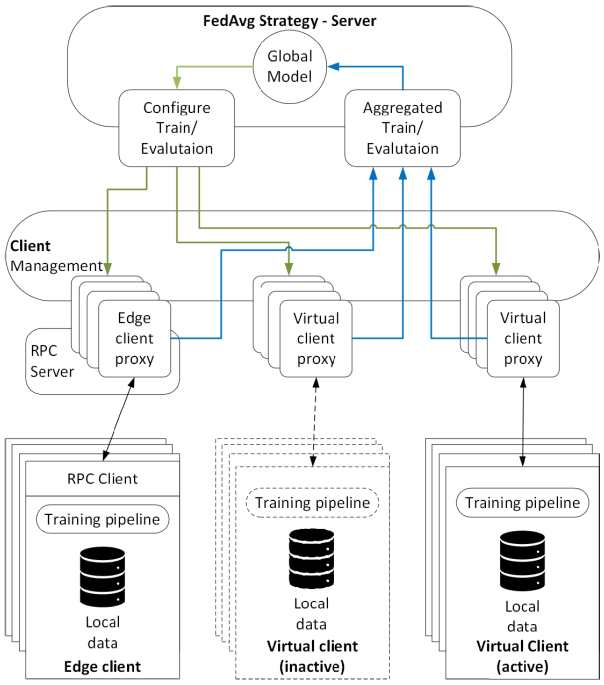

FIGURE 6. Data exchange in federated learning mechanism of FlwrBC.

configure_evaluate/aggregate_evaluate enables strategies to perform server-side and client-side evaluations.

The framework principle focuses on the communication between clients and the server, as shown in Figure 6. The server is responsible for coordinating the federated learning process on the available set of client machines. Clients compute the local training of the model using their own data. Global logic concerning the selection of clients, configuration of the global model being trained, and the algorithm for aggregating the model's parameters (aggregated weights) is defined in the abstract strategy class. This study employs a federated averaging algorithm. This strategy is called the FedAvg Strategy. Local logic is defined for training and evaluating local data. The server has three main components: client management for client proxies, federated learning loops, and strategies. A client proxy represents a client connection to the server, and is responsible for transmitting messages to the client. The federated learning loop relies on the strategy class to configure the next round of learning, and sends that configuration to available clients while also receiving parameters from the clients, the state of the clients (success or failure), and the aggregated parameters of the global model.

The workflow of the machine learning component is shown in Algorithm 1. First, in the session initialization step, the server initializes the parameter set w^0 from the global model of the previous session or selects a random client parameter set w^0 if this is the first session and passes it to the participating clients in that round. In addition to w^0, the parameters for the number of rounds, local epochs, batch size, and global model parameter aggregation algorithm were also defined by the server in this step. Second, in step two, depending on the

Algorithm 1 FlwrBC – Federated Learning Process

Input:
Strategy parameters
Set of clients participating in the training: $C = \{C_k; k = 1, 2, \ldots, n\}$
The number of rounds in the session: R
Output:
The global weights of the session

Step 1: initialize
Server side
initialize strategy parameters
if w_0 exist from previous session:
Broadcast w_0 to C
else:
take w_0 from a random client and broadcast to C
Step 2: local training
Each client $\mathbf{C_k} \in C$ **execute**
C_k load global model architecture
C_k pre-process dataset P_k
$r \leftarrow 1$
while $r \leq R$:
C_k training on P_k
send w_k^r to server
Step 3: aggregated model
server receives local models from C, and aggregated model for next round by FedAvg
$w^r \leftarrow FedAvg(w_k^r, k = 1, 2, \ldots, n)$
validate model with parameter w^r
send w^r to C for next round
$r \leftarrow r + 1$
Step 4: repeat until $\mathbf{r = R}$ **or model is converged**
return global model

number of rounds R specified in the initialization step, clients train locally on their own dataset based on the initialization parameters. Subsequently, client k sends its local model parameter w_k^r at round r back to the server for aggregation. In step three, after aggregating the models using the FedAvg algorithm, the server evaluated the global model w^r at round r on the test dataset. In step four, the server sends the global model parameter w^r of that round back to clients to continue the training process. This cycle was repeated until a specified number of rounds ($r = R$) is completed.

C. THE MECHANISM OF BLOCKCHAIN IN FLWRBC

1) SMART CONTRACTS IN FLWRBC

Owing to the decentralized, immutable, secure, and transparent characteristics of the blockchain, the framework utilizes blockchain technology to record federated learning activities and incentivizes clients to participate in federated learning. As outlined in the general framework, the blockchain in the

Algorithm 2 Contribution Contract Interface

```
Procedure CALCULATECONTRIBUTION
    Check whether a Work was not
contributed
    Check whether a Work finished
    if a Work is finished and has
proper dataSize then create a block for
contribution and a block for reward
      emit contributeEvent
   end if
end procedure

function GETCLIENTADDRESSES: returns an
array of client addresses
function GET_RNOS: returns an array of
rNos
function GETCONTRIBUTION: returns
contribution
function GETBALANCE: returns balance
```

system consists of two smart contracts: the Federation Contract (see Algorithm 3), which is responsible for storing and tracing the global model aggregation activities at the server; and the Contribution Contract (see Algorithm 2), which is used to store, trace, and reward the local training activities of clients. Algorithms 2 and 3 define the interfaces between these two smart contracts.

2) REWARDING MECHANISM

According to a study by Yu [31], continuous client participation and long-term federated learning are essential for achieving high model performance. Some famous frameworks of federated learning, such as TensorFlow Federated (TFF) or FATE by WeBank, assume that there is always an existing participating group to join the process; therefore, those frameworks want reward mechanisms. However, this assumption was impractical.

In the same study, Yu identified three common approaches to profit sharing: (1) egalitarian profit sharing, where profits are evenly distributed based on the amount of data; (2) utilitarian profit sharing, where profits are distributed based on the marginal utility gained from each client's participation; and (3) loss-based profit sharing, where profits are determined by the reduction in utility due to a client's failure to participate.

In this study, we developed a reward mechanism based on egalitarian profit-sharing by calculating the reward based on the client's data size and the number of rounds participating in a training session. Specifically, in a training session with a predefined budget B, R rounds, and N participating clients, client C_i with dataset d_i and participating in r_i rounds ($r_i \leq R$) receives a reward P_i as shown in Equation 1.

$$P_i = \frac{d_i * r_i}{\sum_{i=1}^{N} d_i} * \frac{B}{R}. \tag{1}$$

Algorithm 3 Federation Contract Interface

```
Declare contract Federation
   Declare struct Weight
     Declare variable dataSize of type uint
     Declare variable filePath of type
string
    Declare variable fileHash of type
string
   End struct

   Declare struct GlobalModel
    Declare variable filePath of type
string
    Declare variable fileHash of type
string
   End struct

   Declare struct Strategy
     Declare variable algoName of type
string
     Declare variable numRounds of type uint
     Declare variable numClients of type
uint
   End struct

     Declare mapping weights with key of
type uint, value of mapping with key of
type uint, and value of mapping with key of
type address to store Weight structs

    Declare mapping models with key of type
uint, value of mapping with key of type
uint to store GlobalModel structs

     Declare mapping strategies with key of
type uint to store Strategy structs

     Declare event addStrategyEvent with
parameters _algoName of type string,
_num_client of type uint, and _num_round
of type uint

     Function addWeight: store weight
information on the blockchain
     Function getWeight: return the weight
of given session and given round number

     Function addModel: store global model
information at the specific session and
round number to the blockchain
    Function getModel: return global model
of a given session and given round number

    Function addStratergy: store strategy
algorithm information at the specific
session to the blockchain
     Function getStratergy: return strategy
algorithm information of a given session
```

IV. EXPERIMENT

A. ENVIRONMENTAL SETTING

The FlwrBC framework was primarily written in Python and built on two main frameworks: Tensorflow and Flower. TensorFlow was used for pre-processing data, building models, and training model functions. A flower is used to deploy federated machine-learning models. For blockchain simulation, FlwrBC uses truffle and ganache to simulate the Ethereum blockchain and deploys smart contracts. The project was written in Python and the NodeJS package was used. Therefore, Anaconda set up a virtual environment with a configuration specified in the ENV.txt file of the project. The test machine had a configuration similar to that of an AMD Ryzen 7 5800H CPU (16 threads, 8 cores, 3.20 GHz), 16GB RAM, GTX1650 4GB GPU, and Windows 11 OS. The system has three main components: the Blockchain, Server, and Client.

B. DATASET DESCRIPTION

In this experiment, FlwrBC was used to perform the image classification using the CIFAR-10 dataset [32]. CIFAR-10 is a popular computer vision dataset used for image-classification problems. The CIFAR-10 dataset contains 60,000 randomly arranged color images (50,000 training images and 10,000 testing images) belonging to ten classes, each with 6,000 images (5,000 training and 1,000 testing). The classes in CIFAR-10 include airplanes, automobiles, birds, cats, deer, dogs, frogs, horses, ships, and trucks. Each image in the CIFAR-10 had a size of 32×32 pixels. We used a convolution neuron network with 3×3 layers to train this dataset for both centralized and federated learning.

C. PERFORMANCE EVALUATIONS

To evaluate the performance and compare local and federated machine learning, the following metrics were used:

- *Accuracy:* The percentage of correctly predicted data from the total number of observed samples is calculated. True Positives (TP) are correctly predicted data belonging to the correct group. False Positives (FP) are incorrectly predicted data belonging to the correct group. True Negatives (TN) are correctly predicted data belonging to the wrong group, and False Negatives (FN) are incorrectly predicted data belonging to the wrong group. The accuracy formula was as follows: accuracy = (TP + TN) / (TP + NP + TN + FN).
- *Loss:* Determines the difference between the model predictions and actual values. In this thesis, the sparse-categorical cross-entropy loss function is used. A lower loss value indicated that the model predictions were closer to reality.
- *Precision:* Calculates the percentage of correctly predicted data in the correct group out of the total predicted data in the correct group. The formula used was Precision = TP / (TP + NP).
- *Recall:* Calculate The percentage of correctly predicted data in the correct group is calculated from the total actual data. The recall formula was Recall = TP / (TP + FN).
- *F1-Score:* a combined score for recall and precision. A higher F1 value indicates a higher accuracy and recognition of a model. The F1-Score formula was F1 = 2 × precision × recall / (precision + recall).
- *ROC-AUC score:* calculated by plotting an ROC curve and calculating the area under the ROC curve based on the ratio between the true positive and false positive rates.

This study compares the accuracy and confusion matrix metrics with the local machine learning results. The training scenario for local machine learning was as follows: batch size,32; epochs,10; dataset, CIFAR-10. The training scenario for federated machine learning was as follows: number of clients,3; local batch size,32; local epochs,2; and number of training rounds,10. Figure 7 shows a comparison of centralized learning and FL in terms of the accuracy and loss metrics. These metrics indicate that both approaches provided models with equal accuracy after training, where centralized learning achieved 75.7% and federated learning reached 76.3%. By contrast, the loss metric between the two models shows that local machine learning has a better index (0.766 for FlwrBC and 0.698 for local machine learning).

However, in Figure 8, when comparing the confusion matrices, the results of the two models show that the FlwrBC model can recognize each class better than Centralized Learning. Notably, in classes 0, 2, 4, and 5, FlwrBC had a darker color than Centralized Learning. Darker-colored cells exhibited high values. If the elements on the main diagonal of the confusion matrix have large values, and the remaining elements have small values, it is considered a good model. In other words, when representing colors, the darker the diagonal line compared to the rest, the better. Similarly, the ROC-AUC score of for each class in FlwrBC outperformed that of local machine learning. Especially in class 3, both models had the lowest ROC-AUC score, but FlwrBC had a higher score of 93.56% compared to 92.16% in centralized learning. This is also in agreement with the comparison of F1 scores in Table 1, where the FlwrBC metric yields higher outcomes than the local machine learning by two percentage points; specifically, the average F1-score of FlwrBC is 0.78 compared to 0.76 of the local machine learning.

The efficiencies of the two machine-learning methods, Centralized Learning and Federated Learning, are compared in Table 1. The table displays the performance of each method in terms of three metrics: Precision, Recall, and F1-Score, which measure how well the methods can classify the data into ten different classes. In addition, the table shows the average performance of each method across all classes. According to the results, Federated Learning is slightly more efficient and robust than Centralized Learning, as it achieves higher values for all three metrics in most classes and average performance. In Classes 4 and 9, Centralized Learning performed similarly or slightly better than Federated Learning.

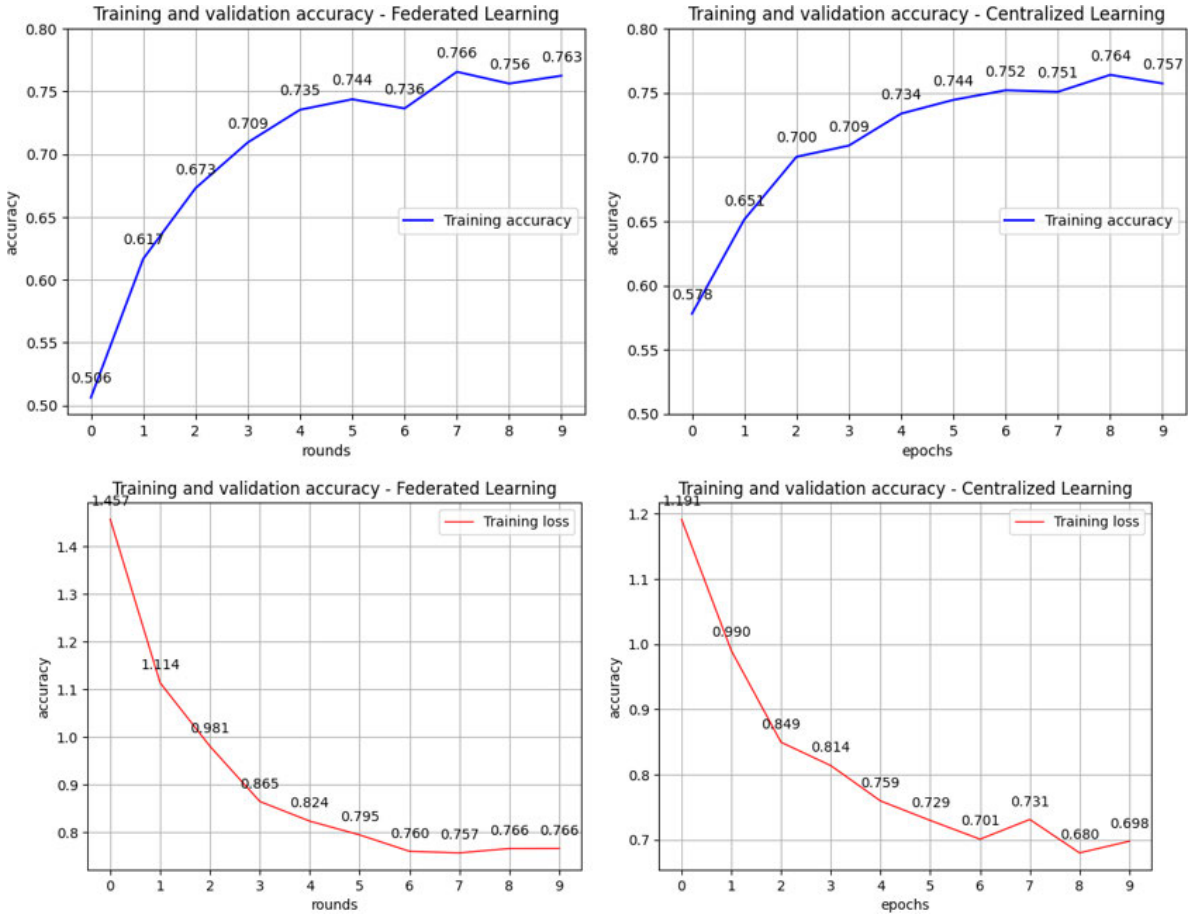

FIGURE 7. Comparison of accuracy and loss between FlwrBC and centralized learning.

TABLE 1. Precision, recall, F1-score between FlwrBC and centralized learning.

Class	Centralized learning			Federated learning		
	Precision	Recall	F1-Score	Precision	Recall	F1-Score
0	0.83	0.63	0.72	0.86	0.73	0.79
1	0.92	0.87	0.89	0.91	0.88	0.89
2	0.68	0.65	0.66	0.67	0.71	0.69
3	0.56	0.59	0.57	0.63	0.57	0.60
4	0.73	0.77	0.75	0.73	0.78	0.75
5	0.69	0.64	0.66	0.65	0.69	0.67
6	0.83	0.85	0.84	0.76	0.87	0.81
7	0.82	0.81	0.81	0.84	0.79	0.81
8	0.73	0.92	0.81	0.84	0.88	0.86
9	0.82	0.88	0.85	0.90	0.86	0.88
avg	**0.76**	**0.76**	**0.76**	**0.78**	**0.78**	**0.78**

Class 4 had identical performances for both methods, with a precision, recall, and F1-score of 0.81. In class 9, Centralized Learning had a slightly higher performance, with a precision of 0.83, recall of 0.82, and F1-score of 0.82, compared to Federated Learning's precision of 0.82, recall of 0.81, and F1-score of 0.81. The most significant difference between the methods was observed in Class 3, where Federated Learning had a precision of 0.88, a recall of 0.87, and an F1-score of 0.87. In contrast, Centralized Learning had a precision of 0.75, recall of 0.74, and F1-score of 0.74. This result indicates that Federated Learning is more accurate and consistent in identifying Class 3 than Centralized Learning. The smallest difference between the methods was observed in class 10, where Federated Learning had a precision of 0.79, recall of 0.78, and F1-score of 0.78, while Centralized Learning had a precision of 0.78, recall of 0.77, and F1-score of 0.77. This

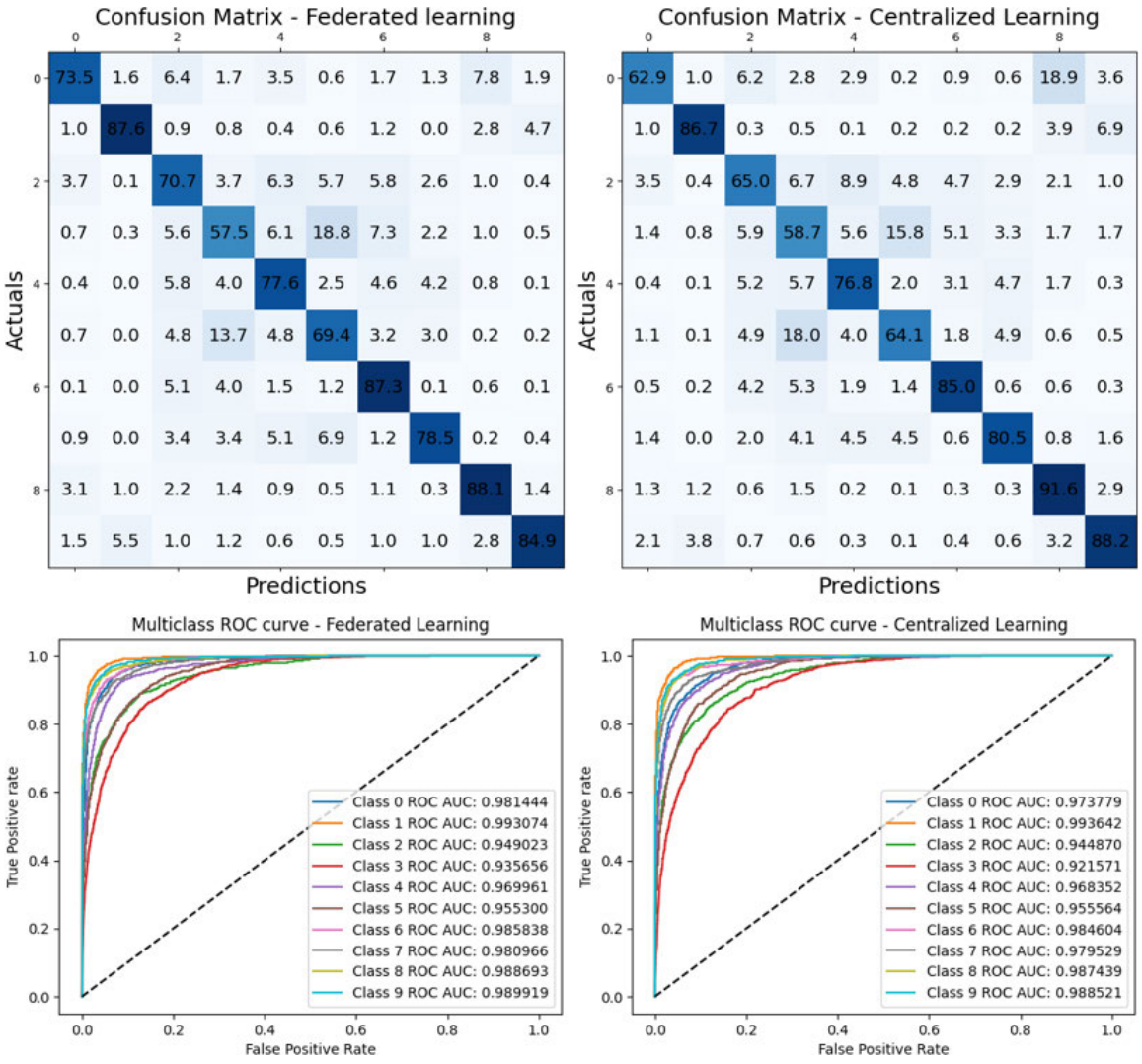

FIGURE 8. Comparison of confusion matrix and ROC curve between FlwrBC and centralized learning.

similarity indicates that both methods were almost equally efficient in classifying Class 10.

V. CONCLUSION AND FUTURE WORKS

As the amount and variety of data grow, security and data privacy pose significant challenges to data mining for machine learning. Traditional machine learning methods often do not address these challenges and require considerable bandwidth to exchange parameters between clients and servers during model training. Federated learning has been proven to offer several benefits over traditional methods, including reduced bandwidth load and improved security and privacy concerns. However, federated learning also has risks such as the risk of lazy clients who do not want to contribute to the model, complex traceability, and single-point failure. Blockchain technology has been integrated to address the abovementioned challenges by leveraging its unique technical features to improve the model performance. The experimental results demonstrate that the performances of both machine learning models are comparable, with federated learning showing slightly better results. Distributing the workload among client machines reduces the server workload, thereby ensuring the efficiency of other tasks.

Although the system currently addresses simple image classification, the framework can be adapted to handle other issues by changing the machine-learning algorithm within the system. Additionally, the system will further explore reward algorithms that focus on the degree of contribution of local models to the global model rather than solely relying on the data size. This new reward system encourages higher-quality data, rather than simply rewarding quantities. Furthermore, the author seeks other blockchain platforms that are more suitable and cost-effective as the system scales up.

This framework can address issues beyond image classification, such as malware detection and anomaly detection [33]. This approach helps prevent the leakage of user information and encourages the community to participate in the

training process, leading to a broader dataset and a more efficient training process. Nonetheless, the system developed in this thesis is still in progress, and there is ample room for improvement.

REFERENCES

[1] X. Xu, X. Liu, Z. Xu, F. Dai, X. Zhang, and L. Qi, "Trust-oriented IoT service placement for smart cities in edge computing," *IEEE Internet Things J.*, vol. 7, no. 5, pp. 4084–4091, May 2020, doi: 10.1109/JIOT.2019.2959124.

[2] M. Patra, R. Thakur, and C. S. R. Murthy, "Improving delay and energy efficiency of vehicular networks using mobile Femto Access Points," *IEEE Trans. Veh. Technol.*, vol. 66, no. 2, pp. 1496–1505, Feb. 2017.

[3] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Int. Conf. Artif. Intell. Statist.*, vol. 54, Apr. 2017, pp. 1273–1282.

[4] Y. Qin, M. Li, and J. Zhu, "Privacy-preserving federated learning framework in multimedia courses recommendation," *Wireless Netw.*, vol. 29, no. 4, pp. 1535–1544, May 2023, doi: 10.1007/s11276-021-02854-1.

[5] X. Li, M. Huang, S. Gao, and Z. Shi, "A fair and verifiable federated learning profit-sharing scheme," *Wireless Netw.*, pp. 1–16, Sep. 2022, doi: 10.1007/s11276-022-03110-w.

[6] X. Li, J. Peng, S. Gao, Z. Shi, and C. Li, "Achieving fair and accountable data trading for educational multimedia data based on blockchain," *Wireless Netw.*, pp. 1–13, Jun. 2022, doi: 10.1007/s11276-022-03042-5.

[7] H. Wang, M. Xiao, C. Wu, and J. Zhang, "Distributed classification for imbalanced big data in distributed environments," *Wireless Netw.*, pp. 1–12, Feb. 2021, doi: 10.1007/s11276-021-02552-y.

[8] A. Mokhtari, W. Shi, Q. Ling, and A. Ribeiro, "Decentralized quadratically approximated alternating direction method of multipliers," 2015, *arXiv:1510.07356*.

[9] C. Zhang, H. Lee, and K. Shin, "Efficient distributed linear classification algorithms via the alternating direction method of multipliers," in *Proc. 15th Int. Conf. Artif. Intell. Statist.*, vol. 22, 2012, pp. 1398–1406.

[10] M. Asad, A. Moustafa, and T. Ito, "Federated learning versus classical machine learning: A convergence comparison," 2021, *arXiv:2107.10976*.

[11] X. Xu, B. Shen, X. Yin, M. R. Khosravi, H. Wu, L. Qi, and S. Wan, "Edge server quantification and placement for offloading social media services in industrial cognitive IoV," *IEEE Trans. Ind. Informat.*, vol. 17, no. 4, pp. 2910–2918, Apr. 2021, doi: 10.1109/TII.2020.2987994.

[12] S. Abdulrahman, H. Tout, H. Ould-Slimane, A. Mourad, C. Talhi, and M. Guizani, "A survey on federated learning: The journey from centralized to distributed on-site learning and beyond," *IEEE Internet Things J.*, vol. 8, no. 7, pp. 5476–5497, Apr. 2021.

[13] L. T. Phong and T. T. Phuong, "Privacy-preserving deep learning via weight transmission," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 11, pp. 3003–3015, Nov. 2019.

[14] J. Konečný, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," 2016, *arXiv:1610.02527*.

[15] A. E. Ouadrhiri and A. Abdelhadi, "Differential privacy for deep and federated learning: A survey," *IEEE Access*, vol. 10, pp. 22359–22380, 2022.

[16] N. Bouacida and P. Mohapatra, "Vulnerabilities in federated learning," *IEEE Access*, vol. 9, pp. 63229–63249, 2021.

[17] W. Lin, X. Yin, S. Wang, and M. R. Khosravi, "A blockchain-enabled decentralized settlement model for IoT data exchange services," *Wireless Netw.*, pp. 1–15, May 2020, doi: 10.1007/s11276-020-02345-9.

[18] C. Zhu, X. Zhu, J. Ren, and T. Qin, "Blockchain-enabled federated learning for UAV edge computing network: Issues and solutions," *IEEE Access*, vol. 10, pp. 56591–56610, 2022.

[19] U. Majeed, L. U. Khan, S. S. Hassan, Z. Han, and C. S. Hong, "FL-Incentivizer: FL-NFT and FL-tokens for federated learning model trading and training," *IEEE Access*, vol. 11, pp. 4381–4399, 2023.

[20] H. A. Madni, R. M. Umer, and G. L. Foresti, "Blockchain-based swarm learning for the mitigation of gradient leakage in federated learning," *IEEE Access*, vol. 11, pp. 16549–16556, 2023.

[21] G. Hua, L. Zhu, J. Wu, C. Shen, L. Zhou, and Q. Lin, "Blockchain-based federated learning for intelligent control in heavy haul railway," *IEEE Access*, vol. 8, pp. 176830–176839, 2020.

[22] M. A. Rahman, M. S. Hossain, M. S. Islam, N. A. Alrajeh, and G. Muhammad, "Secure and provenance enhanced Internet of Health Things framework: A blockchain managed federated learning approach," *IEEE Access*, vol. 8, pp. 205071–205087, 2020.

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

[24] F. Yu, H. Lin, X. Wang, A. Yassine, and M. S. Hossain, "Blockchain-empowered secure federated learning system: Architecture and applications," *Comput. Commun.*, vol. 196, pp. 55–65, Dec. 2022.

[25] M. Osmanoğlu and A. A. Selçuk, "Privacy in blockchain systems," *Turkish J. Electr. Eng. Comput. Sci.*, vol. 30, no. 2, pp. 344–360, 2022.

[26] B. Bünz, S. Agrawal, M. Zamani, and D. Boneh, "Zether: Towards privacy in a smart contract world," in *Financial Cryptography and Data Security*, vol. 12059. Cham, Switzerland: Springer, Jul. 2020.

[27] R. Kumar and R. Tripathi, "Implementation of distributed file storage and access framework using IPFS and blockchain," in *Proc. 5th Int. Conf. Image Inf. Process. (ICIIP)*, Nov. 2019, pp. 246–251, doi: 10.1109/ICIIP47207.2019.8985677.

[28] M. Alessi, A. Camillo, E. Giangreco, M. Matera, S. Pino, and D. Storelli, "Make users own their data: A decentralized personal data store prototype based on Ethereum and IPFS," in *Proc. 3rd Int. Conf. Smart Sustain. Technol. (SpliTech)*, Jun. 2018, pp. 1–7.

[29] S. Chaudjary, R. Kakkar, R. Gupta, S. Tanwar, S. Agrawal, and R. Sharma, "Blockchain and federated learning-based security solutions for telesurgery system: A comprehensive review," *Turkish J. Electr. Eng. Comput. Sci.*, vol. 30, no. 7, pp. 2446–2488, 2022.

[30] D. J. Beutel, T. Topal, A. Mathur, X. Qiu, J. Fernandez-Marques, Y. Gao, L. Sani, K. Hei Li, T. Parcollet, P. P. B. de Gusmão, and N. D. Lane, "Flower: A friendly federated learning research framework," 2020, *arXiv:2007.14390*.

[31] H. Yu, Z. Liu, Y. Liu, T. Chen, M. Cong, X. Weng, D. Niyato, and Q. Yang, "A fairness-aware incentive scheme for federated learning," in *Proc. Assoc. Comput. Machinery Conf. AI*, 2020, pp. 393–399.

[32] A. Krizhevsky, V. Nair, G. Hinton. (Jun. 29, 2020). *The CIFAR-10 Dataset*. Accessed: Dec. 7, 2022. [Online]. Available: https://www.cs.toronto.edu/~kriz/cifar.html

[33] D. Saraswat, A. Verma, P. Bhattacharya, S. Tanwar, G. Sharma, P. N. Bokoro, and R. Sharma, "Blockchain-based federated learning in UAVs beyond 5G networks: A solution taxonomy and future directions," *IEEE Access*, vol. 10, pp. 33154–33182, 2022.

NGUYEN TAN CAM received the bachelor's degree in information technology and the master's degree in information systems from the University of Natural Sciences, Vietnam National University Ho Chi Minh City, in 2006 and 2010, respectively, and the Ph.D. degree in information technology from the University of Information Technology, Vietnam National University Ho Chi Minh City (UIT VNU-HCM). He is currently a Lecturer with the UIT VNU-HCM. His main research interests include mobile security, distributed network security, the IoT security, and machine learning-based cyber security.

VU TUAN KIET received the bachelor's degree in information technology from the University of Information Technology, Vietnam National University Ho Chi Minh City (UIT VNU-HCM), in 2023. He is currently a Researcher with the UIT VNU-HCM. His main research interests include malware detection, network security, the IoT security, and machine learning-based cyber security.