future internet

MDPI

Article

pFedBASC: Personalized Federated Learning with Blockchain-Assisted Semi-Centralized Framework

Yu Zhang [1,2], Xiaowei Peng [2] and Hequn Xian [1,2,*]

[1] College of Computer Science and Technology, Qingdao University, Qingdao 266000, China
[2] Cryptography and Cyberspace Security (Whampoa) Academy, Guangzhou 510000, China
* Correspondence: xianhq@qdu.edu.cn

Abstract: As network technology advances, there is an increasing need for a trusted new-generation information management system. Blockchain technology provides a decentralized, transparent, and tamper-proof foundation. Meanwhile, data islands have become a significant obstacle for machine learning applications. Although federated learning (FL) ensures data privacy protection, server-side security concerns persist. Traditional methods have employed a blockchain system in FL frameworks to maintain a tamper-proof global model database. In this context, we propose a novel personalized federated learning (pFL) with blockchain-assisted semi-centralized framework, pFedBASC. This approach, tailored for the Internet of Things (IoT) scenarios, constructs a semi-centralized IoT structure and utilizes trusted network connections to support FL. We concentrate on designing the aggregation process and FL algorithm, as well as the block structure. To address data heterogeneity and communication costs, we propose a pFL method called FedHype. In this method, each client is assigned a compact hypernetwork (HN) alongside a normal target network (TN) whose parameters are generated by the HN. Clients pull together other clients' HNs for local aggregation to personalize their TNs, reducing communication costs. Furthermore, FedHype can be integrated with other existing algorithms, enhancing its functionality. Experimental results reveal that pFedBASC effectively tackles data heterogeneity issues while maintaining positive accuracy, communication efficiency, and robustness.

Keywords: blockchain; semi-centralized framework; personalized federated learning; hypernetwork; non-iid data

Citation: Zhang, Y.; Peng, X.; Xian, H. pFedBASC: Personalized Federated Learning with Blockchain-Assisted Semi-Centralized Framework. *Future Internet* **2024**, *16*, 164. https://doi.org/10.3390/fi16050164

Received: 9 April 2024
Revised: 5 May 2024
Accepted: 10 May 2024
Published: 11 May 2024

1. Introduction

In recent years, the digital era has experienced continuous growth, leading to increased interactions among internet users. This surge in data generated by electronic devices presents significant challenges to data security. Consequently, there is an imperative need to establish a new generation of reliable information management systems. These systems enhance the efficiency of societal operations, lower collaboration costs, and wield substantial influence on economic growth and engineering research.

With the enforcement of General Data Protection Regulation (GDPR), data aggregation has become increasingly complex. This complexity makes it difficult for individual developers to obtain large volumes of high-quality data, resulting in a critical issue of data islands. As an emergent machine learning technique, federated learning (FL) [1,2] enables multiple devices and users to collaboratively develop models in a distributed data storage environment. FL overcomes the data islands problem by offering a cooperative learning approach for machine learning and establishing neural network models that protect data privacy.

Blockchain combines several technologies, including distributed networks, encryption, and smart contracts, to create a trust-based data management infrastructure. In a blockchain, participants collaboratively confirm transactions, and the information is stored in an ever-expanding chain-like structure. Blockchain's decentralization reduces

reliance on trusted third parties. Transaction transparency allows all participants to view and verify transactions, while tamper-proofing ensures the authenticity and reliability of transaction information.

In practical situations, acquiring a trusted server can be challenging. Blockchain satisfies requirements for decentralized, transparent, and tamper-proof data storage, catering to the needs of a decentralized FL. Consequently, blockchain-based federated learning (BFL) has emerged as a novel approach to distributed network data computation. There are several studies in BFL frameworks, with a primary focus on client selection [3,4], aggregation weights [5,6], local training adjustment [5,7], privacy preservation [8–11], and model compression [12]. Additionally, these studies explore consensus mechanisms [3,4,13], block structure [13–15], committee selection [6,15], and incentive mechanism design for blockchain [8–10]. The papers [3–16] identify several issues: the bottom-level FL has not fully exploited the potential of the blockchain, and the top-level blockchain cannot provide comprehensive support for FL. Specifically, the existing BFL framework does not fully take advantage of all network connections in real-world Internet of Things (IoT) situations, leading to high communication costs and low training efficiency. Additionally, the lack of a logical and comprehensive block structure design does not ensure sufficient protection and maintenance of users' intellectual property.

FL also faces challenges such as high communication costs and data heterogeneity. With the increase in device computation capabilities and model complexity, user devices are required to transmit a large number of parameters in each training round. Simultaneously, data heterogeneity issues stemming from user behavior, regional differences, and variations in data size between individuals and organizations affect model accuracy and even convergence.

In the statistical heterogeneity situation, a single global model is not suitable for highly non-independent and identically distributed (non-iid) data. Training using model decoupling methods still involves transmitting all feature knowledge to users subjectively, without providing personalized knowledge specific to local users. To address these issues, we adopt personalized federated learning (pFL), in which a personalized model is learned for every client. Typically, the final personalized models outperform global models that are trained using normal FL techniques.

We propose a new *personalized federated learning with blockchain-assisted semi-centralized framework (pFedBASC)*. This framework focuses on real-world applications, utilizing various types of network connections for collaborative training and leveraging the blockchain to provide comprehensive distributed training support for FL. The pFedBASC aims to minimize communication costs and increase flexibility. The proposed framework allocates aggregation tasks to clients, reduces aggregation time through trusted client connections, and applies a blockchain-assisted approach to optimize the aggregation of untrusted client models, ensuring the local model's adaptability and performance.

We embed a new algorithm, *personalized federated learning with hypernetwork (FedHype)*, into pFedBASC. FedHype is designed to address the challenges of pFL training and data heterogeneity. FedHype directs each client to train a small hypernetwork (HN) and a larger local target network (TN) whose model parameters are generated by the small HN. Clients are permitted to perform local aggregation of HN to train a personalized TN. Only the compact local HN is involved in knowledge exchange among clients, which helps mitigate data heterogeneity and reduces communication costs. pFedBASC can utilize FedHype to solve personalized training issues effectively in BFL.

The main contributions of this paper are as follows:

- We propose a blockchain-assisted semi-centralized pFL framework called pFedBASC, which provides a reliable collaborative training environment for distributed data in IoT and offers guidance for designing BFL algorithms.
- We design the block structure of the blockchain for pFedBASC. We model and formally describe the semi-centralized framework. Building upon this, we design an FL aggregation method, utilizing loss functions and delayed rounds for weight adjustment.

- We propose a pFL algorithm called FedHype, which takes advantage of the hypernetwork's characteristics. FedHype significantly enhances the overall pFL performance and meets the personalized training needs of different users. We also integrate FedHype with other existing algorithms, further extending its functionality.

2. Related Work

2.1. Blockchain-Based Federated Learning

In recent years, the study of BFL frameworks has seen remarkable growth, drawing extensive research interest due to their auditable training processes and serverless architectures, which avoid the single point of failure typical in FL systems. Traditional BFL methods replace the original server with a blockchain, increasing generality and security. For instance, the paper [16] introduces a multi-layer BFL framework, where the upper layer is responsible for global model management, and the lower layer focuses on resource scheduling and updating local models. By utilizing a blockchain, a blended framework of a blockchain and FL has been proposed to manage security and trust issues when applying FL in mobile edge networks. Furthermore, Ref. [12] employs a top-k model compression mechanism in the centralized BFL framework to improve the overall performance, while [14] introduces an autonomous BFL design for privacy-aware and efficient vehicular communication networks that theoretically optimizes Proof of Work (PoW), reducing the latency of BFL.

In research focusing on blockchain mechanisms in BFL, particularly regarding security against malicious attacks on global models or user privacy data, BFLC [15] designs committee mechanisms and block structures to assist in model storage and downloading and innovatively reduces the computation required for consensus and the potential for malicious attacks. FGFL [6] calculates client trustworthiness within the blockchain committee for election, determines aggregation weights based on client contributions, and enhances the system's convergence and effectiveness. In terms of reward mechanisms, FedTwin [8] relies on a centralized BFL framework and utilizes a Generative Adversarial Network (GAN) to ensure privacy preservation, establishing a reward mechanism based on model performance. Additionally, PF-PoFL [9] addresses the drawbacks of PoW and proposes a novel energy-recycling consensus mechanism that utilizes computational power wasted on difficult but meaningless PoW puzzles for practical FL tasks.

The above introduces various examples of blockchain applications in FL. We also apply the blockchain to FL scenarios that we need to improve efficiency and security. Current BFL frameworks primarily focus on optimization and improvement within FL or the blockchain separately. Our pFedBASC emphasizes the collaborative design of both the blockchain and FL. We will introduce a method to support semi-centralized FL training throughout the entire process starting from the blockchain design.

2.2. Personalized Federated Learning

PFL is a method to reduce the effects of data heterogeneity [17]. It maintains different models on different clients, achieving higher accuracy on their respective datasets. Previous methods in pFL encompass fine-tuning [18,19], federated meta-learning [20], federated multi-task learning [21,22], model mixup [23], and federated clustering [24,25]. Recent pFL methods involve FedBN [26], which similar to FedAvg [2], keeps batch normalization layers local. FedPer [27] divides the model into a global layer and a personalized layer, with only the global layer being aggregated. FedRep [23] builds upon FedPer by further analyzing the effectiveness and rationality of local training and separately trains global and personalized layers. FedBABU [28] adopts a similar neural network partitioning approach by aggregating only the global layer. The difference lies in that FedBABU does not train the personalized layer but fine-tunes it after training has been completed. FedFomo [29] enables each client to obtain the complete models from other clients and choose those that offer greater advantages for local aggregation to update their local model, but the communication cost increases exponentially.

Paper [30] presented the fast weighting concept in hypernetworks, wherein one network can generate context-dependent weight adjustments for another network. The first attempt to apply hypernetworks to pFL was in pFedHN [31]. The pFedHN deploys a hypernetwork on the server that is larger than the local target model. The hypernetwork acquires parameters for local models and creates personalized models for each client, though the communication expenses are approximately equivalent to those of FedAvg. Paper [32] implements separate hypernetworks for each user on the server. Additionally, during the model aggregation phase, the server assesses the significance of each layer across various networks. Paper [33] applies the hypernetwork to the local deployment and trains the hypernetwork, local personalization layer, and embedding vector, respectively, reducing communication overhead and increasing accuracy. In the Fed-RoD [34], each client's architecture includes a feature extractor and two separate headers. The global header collaborates with the feature extractor to aggregate data and update models, while a personalized header is produced by the local hypernetwork.

3. Preliminaries

3.1. Semi-Centralized Federated Learning Framework

Traditional BFL frameworks mainly replace the centralized server in FL with a blockchain, functioning similarly to centralized FL. The blockchain's core framework consists of a distributed system and a chain-based database. This database stores block information, with each block containing data, a timestamp, and the previous block's hash value. Block data can take various forms, such as transaction records or a contract code, while timestamps indicate the block creation time. The hash value of the previous block connects the current block to all prior blocks, forming a tamper-proof chain structure. Ensuring credibility by replacing the server side with a blockchain will increase the time for aggregation and thus reduce the overall training efficiency.

In centralized FL, synchronous aggregation is commonly used, causing faster clients to wait for slower clients to complete training before aggregation. In contrast, decentralized FL replaces client–server with peer-to-peer (P2P) communication among clients. In this paper, we consider real-world applications, particularly IoT based on 5G/6G technologies, where clients are interconnected, essentially creating a purely distributed framework. However, when using a decentralized FL, not all connections are available due to their credibility and network latency. Clients may not establish a P2P data transfer if they lack mutual trust or are outside each other's direct connection range, reducing the training efficiency and accuracy. In this situation, we adopt a semi-centralized BFL framework, as shown in Figure 1.

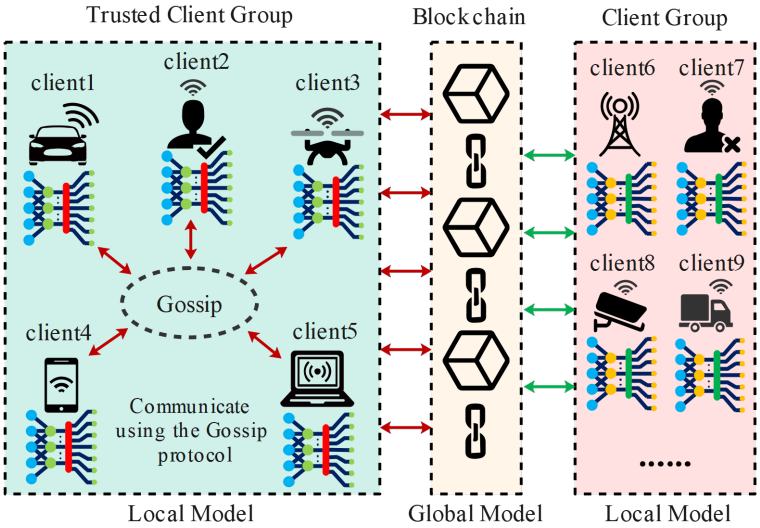


Figure 1. The semi-centralized BFL framework based on trusted client groups.

It integrates centralized and decentralized FL frameworks. We employ the decentralized framework to enhance BFL's transmission efficiency and reduce computation on the blockchain while leveraging the high credibility and connectivity of BFL framework to improve client accuracy and address the data islands problem. Within the trusted client group, clients communicate with each other using the Gossip protocol, directly exchanging model knowledge, which reduces communication costs with the blockchain. For clients without trusted partners, they can also register on the blockchain and collaborate in FL through the blockchain.

3.2. Hypernetwork

The hypernetworks (HNs) [35] belong to the category of generative networks that generate parameters for the target model (TN). The output of a HN is the parameters of TN, which vary based on the input (representing the embedding vector of TN). HN and TN are trained end-to-end, and this process is carried out entirely at the user's local client. The knowledge transfer process in HN involves the following steps: (1) HN generates model parameters for TN, transmitting knowledge in the form of parameter distribution from HN to TN; (2) TN directly trains on the local dataset; (3) HN's parameters are updated using the variances in TN's parameters before and after training, transferring the latest TN's knowledge to HN. This process is repeated until the target model converges. Essentially, HN acts as a knowledge medium, continuously transferring the parameter distribution of TN during user and server training. The structure of HN and TN is shown in Figure 2. With an output dimension of 400, our HN is smaller than the TN's parameter capacity. Consequently, HN is invoked multiple times in a stacked manner to generate parameters. HN input is represented as the embedded vector that combines client id and chunk id. Every 400 parameters form a chunk, with ids sequentially increasing from 0 and incremented by 1.

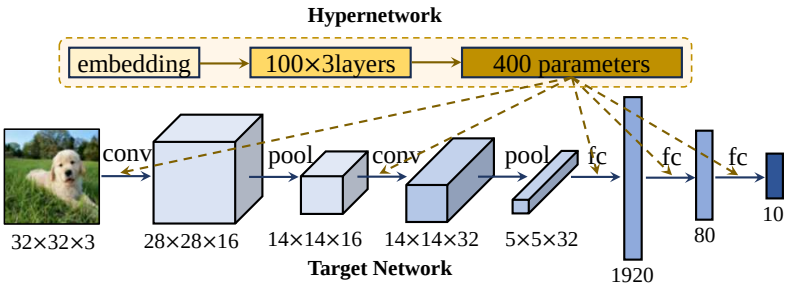


Figure 2. The structure of a hypernetwork and its generated target network.

3.3. Problem Definition

In traditional processes, FL consists of clients C_i where $i \in \{1, 2, ..., N\}$ and a server responsible for aggregating parameters. Each C_i possesses a private sample dataset $\mathcal{D}_i$. The objective of FL is to cooperatively solve an optimization problem. The objective function [2] is defined as follows:

$$\underset{w}{\operatorname{argmin}} \left[F(\boldsymbol{w}) := \sum_{i=1}^{N} p_i F_i(\boldsymbol{w}) \right] \tag{1}$$

where $\boldsymbol{w}$ denotes the model parameters and p_i represents the weight of C_i participating in aggregation, typically based on the rate of C_i data size in training. $F_i(\boldsymbol{w})$ represents the local optimization objective function of C_i, defined as follows:

$$F_i(\boldsymbol{w}) = \mathbb{E}_{(x,y)\in\mathcal{D}_i} L(\boldsymbol{w}; (x, y)) \tag{2}$$

where $L(\boldsymbol{w}; (x, y))$ is the loss function for each client. PFL allows each user to build personalized model to better adapt to local data features. Therefore, the goal of solving pFL in this paper can be transformed into proposing an automated method for generating model summaries to adaptively complete FL training under external conditions. We use the input

embedding vector of HN to describe heterogeneous features of user data. These embedding vectors act as representation vectors, capturing the characteristics and differences between different user entities. For simplicity, we use $h(\cdot)$ to denote model generation operation. In the t-th round, the HN of C_i is $\boldsymbol{\varphi}_i^t$, and the TN is $\boldsymbol{w}_i^t$. Following [31], we can use $\boldsymbol{\varphi}_i^t$ and the *unique* embedding vector $\boldsymbol{v}_i$ to generate the parameters $\boldsymbol{w}_i^t$ through the process (forward propagation):

$$\boldsymbol{w}_i^t = h(\boldsymbol{v}_i; \boldsymbol{\varphi}_i^t) \tag{3}$$

For convenience, $\boldsymbol{v}_i$ represents a set of embedding vectors for all features of C_i, with each embedding vector having 400 parameters. Since HN of each client uses different embedding vectors, the generated TNs are personalized. After generating a TN, we train it on the local dataset to obtain the latest TN $\hat{\boldsymbol{w}}_i^t$, which will be used to update the HN. Following [31], we can use the locally trained TN to perform a one-shot update (backward propagation) of HN, expressed as follows:

$$\boldsymbol{\varphi}_i^t \leftarrow \boldsymbol{\varphi}_i^{t-1} - \eta \, \nabla_{\boldsymbol{\varphi}_i^{t-1}} \left(\hat{\boldsymbol{w}}_i^{t-1} \right)^T \Delta \boldsymbol{w}_i^{t-1} \tag{4}$$

where $\nabla_{\boldsymbol{\varphi}_i^{t-1}} \left(\hat{\boldsymbol{w}}_i^{t-1} \right)^T$ is the derivative of the latest trained TN $\hat{\boldsymbol{w}}_i^{t-1}$ with respect to the HN $\boldsymbol{\varphi}_i^{t-1}$ in the $(t-1)$-th round, and $\Delta \boldsymbol{w}_i^{t-1} = \hat{\boldsymbol{w}}_i^{t-1} - \boldsymbol{w}_i^{t-1}$ represents the variation of generated $\boldsymbol{w}_i^{t-1}$ before and after executing local training. Based on the above setup, the objective for pFL is adjusted to

$$\underset{\boldsymbol{\varphi}}{\operatorname{argmin}} \left[\sum_{i=1}^{N} p_i F_i(h(\boldsymbol{v}_i; \boldsymbol{\varphi})) \right] \tag{5}$$

The logic behind our training approach is as follows: we use the embedding vectors to represent local features of user's data, iteratively update HN parameters, and finally obtain a global optimal pFL model set $\boldsymbol{\varphi}^*$ and $\boldsymbol{w}^* = \{\boldsymbol{w}_1, \boldsymbol{w}_2, ..., \boldsymbol{w}_N\}$ that satisfies the training goal.

4. Methodology

4.1. pFedBASC Framework Overview

We combine centralized and decentralized FL methods to propose a more realistic semi-centralized FL framework, pFedBASC. The overall design of pFedBASC includes designs in both FL methods and the blockchain framework. In the FL part, the main components include the design of the aggregation process and pFL algorithm, FedHype.

During the aggregation process, pFedBASC adopts a local aggregation mode, allowing each client to perform global aggregation individually, which significantly reduces the computation burden on the blockchain. The aggregation among trusted client group members uses the Gossip protocol, which utilizes P2P connections to improve model transmission and aggregation efficiency. Additionally, pFedBASC involves a loss-based weighting aggregation scheme that determines aggregation weights according to model performance, thus enhancing model accuracy. The framework also employs a delayed round based weighting aggregation scheme, which adjusts weights based on the rounds of different local models, preventing outdated parameters from affecting accuracy. Regarding the design of FedHype, the process will be detailed in the following section. FedHype uses HN to replace TN for parameter transfer and aggregation.

In the blockchain design part, we focus on the block structure design to support training. From the block-type perspective, we primarily design the *upload block*, *download block*, and *evaluation block*. In terms of block attributes, we design *header information*, *model information*, and *validation information*.

4.2. Block Structure Design

The design of all blocks in pFedBASC includes three parts: *header information*, *model information*, and *validation information* as shown in Figure 3.

- *Header information*: The block header primarily records information necessary for blockchain's historical records and traceability. This section consists of three standardized pieces of information: the block ID, which provides a unique identifier for the current block, allowing other blocks to distinguish and reference it; the timestamp, which records the time when the block was added to the chain, facilitating subsequent tracebacks and timeline construction for any anomalies; and the block type, which indicates whether the block is a *download*, *evaluation*, or *upload block*, aiding in the subsequent processing of block-specific information. Additionally, the client ID records the identification information of the client involved in the operation, facilitating the traceability of the actor.
- *Model information*: This part contains the information and attributes necessary for the FL training process. It is strongly related to the block type, and detailed descriptions will follow based on the type of block involved.
- *Validation information*: The block includes data crucial for verifying the accuracy of the content within the blockchain. By recording the previous hash of the parent block and the current block's hash, the blockchain's tamper-proof nature is ensured, preventing attackers from compromising the integrity of the blockchain by altering a single block.

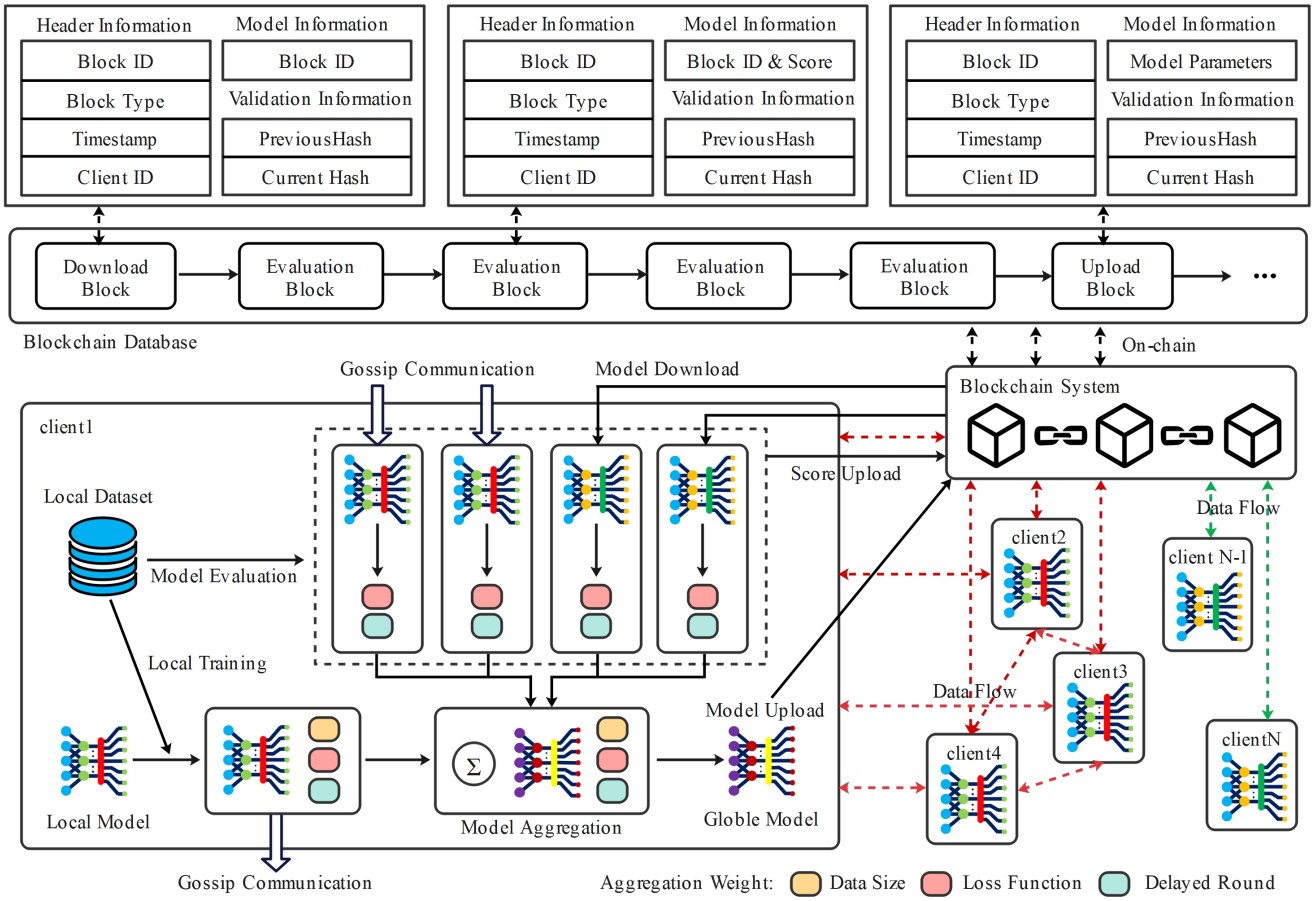


Figure 3. The training procedure of pFedBASC and the design of the block structure.

Based on the above block design, certain personalized designs were made for blocks with different functions to support the designated functions of the blocks.

- *Download Block*: Blocks record the model parameters that a client downloads from the blockchain system. This enables the system to update the contribution score of the model's uploader or provide rewards based on intellectual property rights. Moreover, suppose a client's local model encounters security issues or errors. In that case, the download records can be traced to determine if the problem originated from a model stored in the blockchain system, thus ensuring traceability. In the *header information* of a download block, the client ID records which client performed the download, facilitating traceability. The *model information* section records which uploaded block's model parameters were downloaded, specifically noting the block ID where the model parameters reside. Recording the block ID instead of the model parameters effectively reduces the storage space required for this block.
- *Evaluation Block*: Blocks are tasked with recording a client's evaluation of model parameters within the blockchain system. Primarily, they log the loss function value derived from model inference on this client, allowing the system to track and record the quality and reliability of a particular model's parameters. This tracking can inform incentives for the clients who uploaded the high-quality model parameters, thus encouraging high-quality training. In the *header information*, the evaluation block's client ID identifies which client performed the evaluation, enhancing traceability. The *model information* section records which uploaded block's model parameters were evaluated, including the block ID and the corresponding scores. This paper primarily designs and discusses evaluation blocks, without delving into their further utilization.
- *Upload Block*: Blocks are designated for storing new rounds of locally aggregated model parameters from a client. They are crucial for providing other clients with reliable model parameters for aggregation, aimed at enhancing the accuracy and generalizability of local models across clients. In the *header information*, the upload block's client ID indicates which client uploaded the model parameters, ensuring traceability. The *model information* section records all the model parameters uploaded during this operation.

These specialized block designs are critical for maintaining the functionality, security, and efficiency of the FL system integrated with blockchain technology. Each type of block plays a strategic role in the overarching process, from initiating updates to validating and consolidating learning outcomes, thereby ensuring the robustness and integrity of the FL process.

4.3. pFedBASC Aggregation Method

In pFedBASC, only HN parameters $\boldsymbol{\varphi}$ are involved in the model transfer. Therefore, we only discuss the aggregation method for $\boldsymbol{\varphi}$. We mainly focus on the design of the aggregation method and weight setting. The pFedBASC performs aggregation on client-side, and local aggregation of C_i includes three parts: the first part is local model $\boldsymbol{\varphi}_i^t$, the second part consists of trusted models $\boldsymbol{\varphi}_j^{t_j}$ from trusted client group C_i^{group}, and the third part involves registered models $\boldsymbol{\varphi}_k^{t_k}$ obtained from the blockchain. Clients aggregate model parameters of these three types locally, generating a global model $\tilde{\boldsymbol{\varphi}}_i^t$ for the next round of training. In pFedBASC, we use the following aggregation formula for each client:

$$\tilde{\boldsymbol{\varphi}}_i^t = p_i^t \boldsymbol{\varphi}_i^t + \sum_{C_j \in C_i^{group}} p_j^t \boldsymbol{\varphi}_j^{t_j} + \sum_{C_k \notin C_i^{group}} p_k^t \boldsymbol{\varphi}_k^{t_k} \tag{6}$$

where the weight p denotes the aggregation weight. We employ the Gossip protocol for model transmission within the trusted client group C_i^{group}. The Gossip protocol shares updated models after C_i has completed local training by broadcasting the new local model $\boldsymbol{\varphi}_i^t$ to trusted C_j. All C_j within group C_i^{group} will repeat this operation to share the model until all connected C_j receive the model updates from other clients in the group. This method can effectively improve the communication efficiency.

In particular, considering the inconsistency of data distribution among clients, we design an aggregation weight adjustment scheme based on the loss function. We use local dataset $\mathcal{D}_i$ to perform inference on the models $\boldsymbol{\varphi}_j^{t_j}$ and $\boldsymbol{\varphi}_k^{t_k}$, incorporating the loss function into the weight definition. That is, we randomly sample a batch of data from the client dataset for inference, approximating the loss function on the global dataset. The weight based on the loss function is defined as follows:

$$p_{1,j}^t = \frac{1}{L(\boldsymbol{\varphi}_j^{t_j}; \mathcal{D}_i)} \qquad p_{1,k}^t = \frac{1}{L(\boldsymbol{\varphi}_k^{t_k}; \mathcal{D}_i)} \tag{7}$$

Simultaneously, we consider the inconsistency of computational capabilities among clients, also known as system heterogeneity. System heterogeneity arises from differences in computing hardware and environments, leading to different time costs for client training. Although the decentralized aggregation scheme and Gossip protocol reduce the waiting time between clients, system heterogeneity can lead to models situated in different rounds during aggregation, as shown below:

$$t-1 \neq t_j,\ t-1 \neq t_k,\ t_k \neq t_j \tag{8}$$

Therefore, it is necessary to compensate and adjust outdated models. This paper designs an outdated model compensation scheme based on delayed rounds. This scheme adjusts the weights according to the difference between $\boldsymbol{\varphi}_j^{t_j}$ and $\boldsymbol{\varphi}_k^{t_k}$ and the current training t-th round. The delayed round-based weight definitions are as follows:

$$p_{2,j}^t = \begin{cases} e^{(t_j - t)}, t_j < t \\ 1\ , t_j \geqslant t \end{cases} \qquad p_{2,k}^t = \begin{cases} e^{(t_k - t)}, t_k < t \\ 1\ , t_k \geqslant t \end{cases} \tag{9}$$

Based on the designs of weight adjustment and compensation methods, the overall aggregation formula of the proposed pFedBASC is shown as follows:

$$\tilde{\boldsymbol{\varphi}}_i^t = \frac{p_i^t p_{1,i}^t}{p^t} \boldsymbol{\varphi}_i^t + \sum_{C_j \in C_i^{group}} \frac{p_j^t p_{1,j}^t p_{2,j}^t}{p^t} \boldsymbol{\varphi}_j^{t_j} + \sum_{C_k \notin C_i^{group}} \frac{p_k^t p_{1,k}^t p_{2,k}^t}{p^t} \boldsymbol{\varphi}_k^{t_k} \tag{10}$$

The weights in the above formula include the normalization process, where the normalization parameters p^t satisfy

$$p^t = p_i^t p_{1,i}^t + \sum_{C_j \in C_i^{group}} p_j^t p_{1,j}^t p_{2,j}^t + \sum_{C_k \notin C_i^{group}} p_k^t p_{1,k}^t p_{2,k} \tag{11}$$

4.4. Proposed FedHype

We will introduce the design of FedHype. We attempt to embed HN mentioned in Figure 2 into FL, using the methods in propagation (3) and (4) for parameter updates to satisfy the pFL training objective in (5). HN takes the 400-parameter embedding vector set $\boldsymbol{v}_i$ that combines client ID and chunk ID as input and output TNs according to the knowledge of HN. During the parameter exchange, FedHype only transmits HN. By aggregating HN, the knowledge and parameter generation abilities can be shared with other users.

As shown in Algorithm 1, the workflow is divided into the aggregator and user parts. After training starts, clients' TN parameters, HN parameters, and embedding vectors $\{\boldsymbol{v}_1, ..., \boldsymbol{v}_N\}$ are initialized. Then, C_i generates $\boldsymbol{w}_i^{t-1}$ using $\boldsymbol{\varphi}^{t-1}$ and $\boldsymbol{v}_i$. Next, C_i carries out local training and updates HN via (4). After completion, HN $\boldsymbol{\varphi}_i^t$ will be sent back to the aggregator, waiting for global aggregation. The aggregator executes the algorithm, obtaining global model $\boldsymbol{\varphi}^t$, and proceeds with model distribution. This process is repeated until the network converges or target rounds are reached. We call it the aggregator because

the aggregation function can be undertaken by a blockchain or user, not solely by a server. FedHype can easily be combined with other FL algorithms to replace aggregation methods, where typically FedAvg is utilized in aggregation by default.

Algorithm 1: FedHype

Input: total communication rounds T, client learning rate α, HN learning rate η, datasets $\{D_1, ..., D_N\}$

Output: $\boldsymbol{\varphi}^T$

Initialize the clients' TN parameters, HN parameters, and embedding vectors $\{\boldsymbol{v}_1, ..., \boldsymbol{v}_N\}$
for each round $t = 1$ to T **do**
    at client $i = 1$ to N **in parallel do**
    $\boldsymbol{w}_i^{t-1} = h(\boldsymbol{v}_i; \boldsymbol{\varphi}^{t-1})$
    ClientUpdate:
        $\hat{\boldsymbol{w}}_i^{t-1} \leftarrow \boldsymbol{w}_i^{t-1} - \alpha \nabla F_i(\boldsymbol{w}_i^{t-1})$
        $\Delta \boldsymbol{w}_i^{t-1} = \hat{\boldsymbol{w}}_i^{t-1} - \boldsymbol{w}_i^{t-1}$
        $\boldsymbol{\varphi}_i^t \leftarrow \boldsymbol{\varphi}^{t-1} - \eta \nabla_{\boldsymbol{\varphi}^{t-1}} \left(\hat{\boldsymbol{w}}_i^{t-1} \right)^T \Delta \boldsymbol{w}_i^{t-1}$
    Communication: send $\boldsymbol{\varphi}_i^t$ to aggregator
    end at client
    at aggregator do
    $\boldsymbol{\varphi}^t \leftarrow$ **ModelAggregation**$(\boldsymbol{\varphi}_i^t)$
    Communication: send $\boldsymbol{\varphi}^t$ to each client C_i
    end at aggregator
end for

We also conducted a qualitative analysis of the algorithm's time complexity. On the client side, each participant executes the following primary operations sequentially: first, local parameter generation via the hypernetwork; second, local training, which typically represents the most significant consumption of computational resources, influenced by the dataset size and model complexity; and third, updating hypernetwork parameters, an operation whose complexity is generally determined by the size of the hypernetwork. On the aggregator side, the complexity primarily depends on the aggregation algorithm and the number of parameters within the hypernetwork. Optimizing any of these factors can effectively reduce the overall consumption of computing resources.

4.5. pFedBASC Workflow

The training procedure of pFedBASC is shown in Figure 3. We will explain the workflow of pFedBASC in combination with Algorithm 2. When users participate in the training for the first time, they need to initialize both the model and blockchain parameters. The framework can use any FL algorithm; in this paper, we use FedHype, where the passed and aggregated parameters are denoted as $\boldsymbol{\varphi}$. After training starts, clients update their models locally and broadcast their new model $\boldsymbol{\varphi}_i^t$ to the trusted client group C_i^{group} via the Gossip protocol. C_i receives updated models $\boldsymbol{\varphi}_j^{t_j}$ from $\forall C_j \in C_j^{group}$, while for $\forall C_k \notin \left(C_i^{group} \cup \{C_i\} \right)$, the latest uploaded models $\boldsymbol{\varphi}_k^{t_k}$ are downloaded from the blockchain. The blockchain constructs a *download block* based on client behavior and proceeds with on-chain processes.

Algorithm 2: pFedBASC

Input: total communication rounds T, HN learning rate η, datasets $\{D_1, ..., D_N\}$, trusted client group $\{C_1^{group}, ..., C_N^{group}\}$
Output: $\tilde{\boldsymbol{\varphi}}^T = \{\tilde{\boldsymbol{\varphi}}_1^T, ..., \tilde{\boldsymbol{\varphi}}_N^T\}$

Initialize both local client and blockchain related parameters
at client $i = 1$ to N **in parallel do**
for each round $t = 1$ to T **do**
    $\boldsymbol{\varphi}_i^t \leftarrow$ **ClientUpdate**$\left(\tilde{\boldsymbol{\varphi}}_i^{t-1}\right)$ via Fedhype
    Communication: broadcast $\boldsymbol{\varphi}_i^t$ to client within C_i^{group} via Gossip protocol
    Communication:
        receive $\boldsymbol{\varphi}_j^{t_j}$ from clients in C_i^{group}
        receive $\boldsymbol{\varphi}_k^{t_k}$ from blockchain
        construct a *download block* and submit it to blockchain for on-chain processes
    Evaluate each received model $\boldsymbol{\varphi}$ and calculate all required weights for $p^t = p_i^t p_{1,i}^t + \sum_{C_j \in C_i^{group}} p_j^t p_{1,j}^t p_{2,j}^t + \sum_{C_k \notin C_i^{group}} p_k^t p_{1,k}^t p_{2,k}^t$
    Communication:
        upload evaluation scores to blockchain
        construct an *evaluation block* with the evaluation scores and submit it to block-chain for on-chain processes
    ModelAggregation: $\tilde{\boldsymbol{\varphi}}_i^t = \frac{p_i^t p_{1,i}^t}{p^t} \boldsymbol{\varphi}_i^t + \sum_{C_j \in C_i^{group}} \frac{p_j^t p_{1,j}^t p_{2,j}^t}{p^t} \boldsymbol{\varphi}_j^{t_j} + \sum_{C_k \notin C_i^{group}} \frac{p_k^t p_{1,k}^t p_{2,k}^t}{p^t} \boldsymbol{\varphi}_k^{t_k}$
    Communication:
        upload the aggregated model $\tilde{\boldsymbol{\varphi}}_i^t$ to blockchain
        construct an *upload block* containing the aggregated model and submit it to blockchain for on-chain processes
end for
end at client

Then, C_i evaluates each received and downloaded model to determine the weights p_j^t and p_k^t. C_i conducts local dataset model inference, determining the weights $p_{1,j}^t$ and $p_{1,k}^t$ by the loss function. C_i records the round of each model and computes delayed round compensation weights $p_{2,j}^t$ and $p_{2,k}^t$. The clients submit the above evaluation scores, such as the loss and distinct model index for the blockchain. The construction of an *evaluation block* and execution of on-chain processes follow. C_i aggregates models to obtain the new round's $\tilde{\boldsymbol{\varphi}}_i^t$ for training and uploads it. The blockchain constructs an *upload block*, and on-chain processes follow accordingly. This iterative process continues until the network converges or reaches the target number of rounds.

We also conducted a qualitative analysis of the algorithm's time complexity. On the client side, firstly, each participant executes local parameter updates using FedHype. Secondly, there is model parameter communication, where each client must access data from all other clients; the complexity of this operation depends on the number of clients and the size of the model parameters. Finally, in terms of blockchain interaction, clients perform parameter evaluation and model aggregation based on the results and must submit various types of blocks (*download*, *evaluation*, *upload blocks*) to the blockchain. The complexity is mainly influenced by the model size and the number of clients. Blockchain operations typically involve network delays and processing times, which are not traditionally included in the time complexity analysis.

5. Experiments

5.1. Experiment Settings

Datasets and models. We use four common datasets: MNIST [36], FMNIST [37], CIFAR-10, and CIFAR-100 [38]. Each dataset is divided into an 80% training set and a 20%

testing set. These datasets are distributed across 50 clients. The training set and testing set on each user have the same data distribution, with no overlap between the training and testing sets. We construct non-iid datasets using the Dirichlet distribution *Dir(0.5)*. We train a LeNet for MNIST and several Convolutional Neural Networks (CNN) for the FMNIST, CIFAR-10, and CIFAR-100 datasets. Each CNN comprises two convolutional layers and three fully-connected layers. The HN used in all three tasks is an MLP model with two hidden layers. To visualize the data heterogeneity among users, we described the heterogeneity for the first three datasets of 10 users using *Dir(0.5)*, as demonstrated in Figure 4. The presence or absence of dots indicates the existence of specific data, while the dot size reflects the quantity of that data.

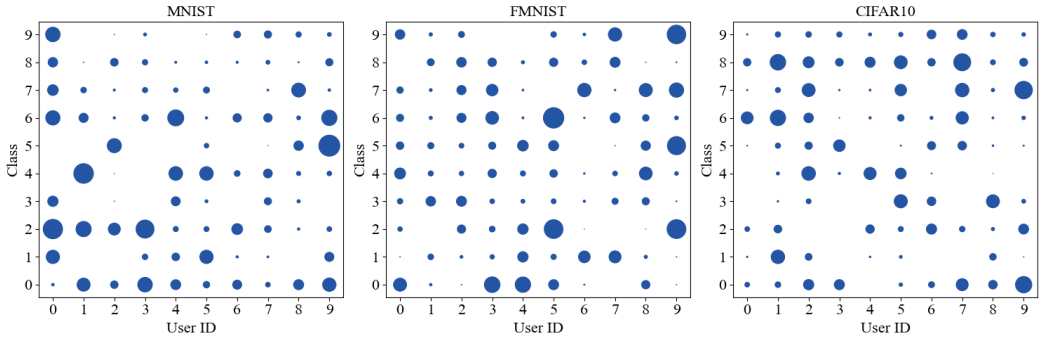


Figure 4. Data distribution of *Dir(0.5)*.

Baselines. We implement the following baseline algorithms: (1) Centralized training, where all data are concentrated for training. This method, while not guaranteeing data security, serves as an upper bound for FL accuracy. (2) FedAvg [2], an important FL algorithm that is effective in various data distributions. (3) Local training, which involves no parameter exchange, and users only use their own data for training. (4) FedProx [39], addressing non-iid data by adding approximation terms to prevent user models from deviating from global model. (5) FedGen [40], which improves FL accuracy by training feature generators. (6) FedBN, addressing FL data heterogeneity problems by adding batch normalization layers to local models. (7) FedBabu [28], an algorithm that updates and aggregates model bodies, requiring fine-tuning after convergence. (8) FedDyn [41], proposing a dynamic regularizer for each user in each round. (9) FedPer [27], which combats adverse effects of statistical heterogeneity by retaining a personalization layer locally. (10) FedRep [23], training classifiers and feature extractors sequentially, only aggregating feature extractors. (11) pFedSim [42], a personalized algorithm based on model similarity. (12) pFedLa [32], deploying a HN on the server-side for each user, granting users tiered aggregation weights. (13) FedFomo [29], providing the best weights for clients to aggregate models by assessing the benefits clients gain from other clients' models. (14) Fed-RoD [34] with HN, involving a feature extractor and two distinct headers for each client.

Settings. Our experiments were conducted on an Ubuntu 18.04 system equipped with an Intel i7-8700K CPU, GTX 1080 Ti GPU, and 16 GB of RAM. We employed Golang for blockchain component development, while PyTorch 1.4.0 was used for developing and training FL network models. Communication between the Go-based blockchain and Python-based models was facilitated using the go-python library. A simulated blockchain network with multiple virtual nodes was run on a single machine, and training data were pre-processed in Python to ensure readiness for model training. We use SGD as the optimizer, setting Nesterov Momentum to 0.9, weight decay to 5×10^{-4}, batch size to 128, the total number of users to 50, global communication rounds for MNIST and FMNIST to 100, and for CIFAR-10 and CIFAR-100 to 150. We conducted several experiments to determine the best learning rates for all algorithms in each experiment. The learning rates considered include multi-step rates decaying from 0.1 and fixed rates of 0.1, 0.01, and

0.001. The optimal learning rate is chosen for all algorithms. In centralized FL, all clients participate in every round of aggregation. The default training process addresses non-iid data heterogeneity but does not include system heterogeneity situations.

5.2. Impact of System Heterogeneity

We conducted an extensive evaluation on CIFAR-10, taking into account system heterogeneity in which we set the computation speed of 50% of clients to be twice as slow as the other 50%. From the perspective of an increased average device training time ratio and decreased device computing runtime ratio, all algorithms are significantly affected by system heterogeneity. As shown in Table 1, this results in an increased training time ratio, compelling devices to wait longer to complete training. In contrast, pFedBASC is less affected by system heterogeneity, with training taking only 1.38 times longer than normal. Considering the device non-idle runtime ratio, normal FL and pFL methods have a ratio below 50%. On the other hand, the framework proposed in this paper achieves a 100% device runtime ratio. Although the aggregation adopted by pFedBASC is distributed and carried out synchronously on various clients, it does not restrict aggregation to only the current round's model. This feature implies that there is no need to wait for other clients to complete training, thus avoiding the wait time and increasing time utilization efficiency. Hence, we observed in the experiment that the overall running time of semi-centralized pFedBASC is not significantly different from other algorithms. This design effectively prevents outdated parameters from affecting aggregation.

Table 1. The experimental results of pFedBASC performance under system heterogeneity and non-system heterogeneity situations on CIFAR-10.

Method	System Heterogeneity		Non-System Heterogeneity		Training Time Increase Ratio
	Accuracy	Device Runtime	Accuracy	Device Runtime	
FedAvg	60.96%	36.67%	63.44%	45.82%	2.06×
FedProx	58.98%	34.45%	63.26%	44.75%	2.31×
FedDyn	59.74%	35.29%	61.33%	47.68%	2.38×
FedGen	57.76%	36.45%	60.85%	49.73%	2.07×
FedBN	60.24%	32.29%	62.91%	42.2%	2.04×
FedBabu	64.01%	36.06%	66.79%	42.59%	2.02×
FedPer	63.54%	36.66%	65.76%	43.9%	1.93×
FedRep	65.6%	32.66%	68.19%	42.64%	2.72×
pFedLa	57.71%	36.91%	60.26%	45.92%	2.25×
pFedBASC	**65.73%**	**100%**	**69.8%**	**100%**	**1.38**×

As system heterogeneity is eliminated, the device computation time ratio increases but does not exceed 50%. This outcome shows that under training tasks and network models in this paper, fluctuations in the device itself lead to significant communication overhead, reducing the system training efficiency and device utilization. This observation further demonstrates the necessity and effectiveness of adopting the semi-centralized FL framework proposed in this paper.

5.3. Performance Evaluation of pFedBASC

Table 2 displays the personalized accuracy for independent and identically distributed (iid) data and non-iid data under the same communication round limit. The performance advantage of pFedBASC is not obvious when starting with the simpler MNIST. This obser-

vation is largely due to the straightforward nature of the MNIST dataset and the relatively simple model structure employed. In such scenarios, even the basic FedAvg suffices to capture most dataset features. Continuing to use HN would introduce unnecessary computational overhead and complicate the training process, potentially deteriorating outcomes. Despite this, it is noteworthy that there is only a small gap between our method and the optimal value method in MNIST. This is because each method can readily accomplish the training task with MNIST, which necessitates further experiments on the remaining datasets. It is also worth noting that HNs were initially designed for more complex datasets and model architectures. The strength of the result on the FMNIST dataset is not consistently apparent, and although pFedBASC demonstrates some advantages in iid data, the difference compared to other methods remains marginal.

Table 2. The experimental results of pFedBASC performance on iid and non-iid datasets across four datasets.

Method	MNIST		FMNIST		CIFAR-10		CIFAR-100	
	non-iid	iid	non-iid	iid	non-iid	iid	non-iid	iid
Central	90.3%		85.38%		76.99%		49.48%	
FedAvg	89.43%	90.02%	83.84%	81.56%	63.44%	56.24%	30.72%	30.62%
Local	86.39%	84.79%	78.72%	69.92%	51.75%	27.59%	18.21%	6.33%
FedProx	89.75%	**90.09%**	84.68%	82.04%	63.26%	55.13%	28.28%	30.94%
FedDyn	89.44%	89.98%	84.76%	82.18%	61.33%	56.05%	26.43%	31.13%
FedGen	88.78%	89.1%	83.54%	80.86%	60.85%	56.76%	23.92%	14.57%
FedBN	**90.09%**	90.07%	82.98%	81.55%	62.91%	30.59%	32.13%	29.14%
FedBabu	89.9%	89.25%	**85.21%**	82.59%	66.79%	58.52%	39.74%	33.3%
FedPer	89.55%	89.62%	83.86%	81.61%	65.76%	57.04%	28.3%	15.53%
FedRep	88.75%	89.11%	83.45%	78.8%	68.19%	59.57%	26.1%	11.2%
pFedSim	89.58%	89.88%	83.78%	82.4%	64.05%	59.48%	39.72%	33.75%
pFedLa	89.59%	89.14%	83.96%	80.49%	60.26%	51.76%	26.73%	23.73%
pFedBASC	88.45%	89.45%	84.54%	**83.19%**	**69.8%**	**70.99%**	**43.6%**	**41.31%**

The advantage of pFedBASC becomes more apparent when utilized in more complex models and datasets (CIFAR series). This is because, in complex tasks, model parameter selection may be more critical, and HN's embedding vector can provide more suitable parameters by adaptively generating higher-accuracy models.

As demonstrated in Table 3, pFedBASC ensures the highest accuracy while maintaining reasonable communication costs, suggesting an optimal balance between model accuracy and communication efficiency. Subsequent experiments revealed that although FedFomo exhibits minimal communication overhead on the CIFAR-10/100 datasets, its accuracy significantly lags behind that of our pFedBASC.

Table 3. Performance of pFedBASC on non-iid CIFAR series datasets: 'Method (xx%)' indicates communication traffic (GB) required to achieve xx% accuracy.

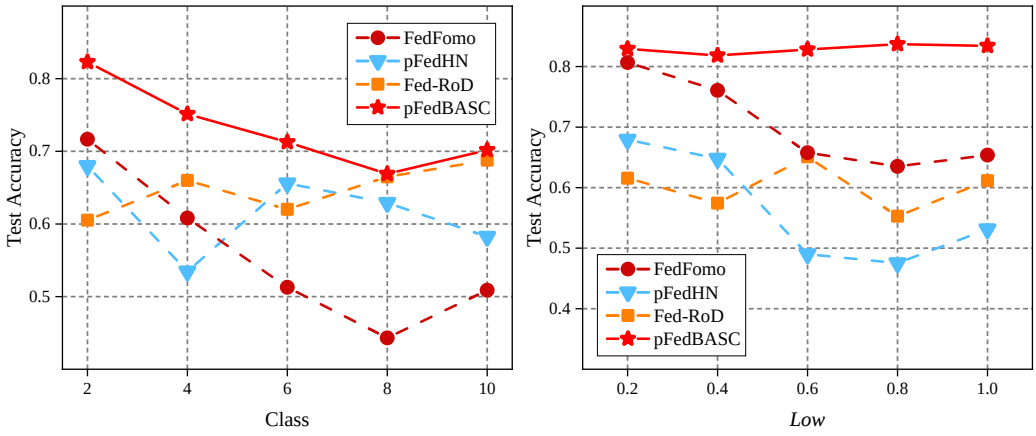
Method	CIFAR-10 (40%)	CIFAR-100 (10%)
FedFomo	1.82	0.52
pFedHN	2.08	10.86
Fed-RoD	9.58	6.17
pFedBASC	**7.97**	**10.62**

5.4. Robustness of pFedBASC

For CIFAR-10 datasets, to construct imbalance, we adjust the quantity of samples per class among various clients. The class sample ratios are determined using a *random.uniform(low,high)* function, where $(low,high \leq 1)$. A greater disparity between the minimum and maximum values leads to a more pronounced imbalance among the classes.

Handling Non-iid Class Numbers. We assign varying numbers of data classes $\{2, 4, 6, 8, 10\}$ to each client. Figure 5a shows that our pFedBASC consistently delivers optimal accuracy on all non-iid datasets, demonstrating its robust performance against non-iid conditions.

Handling Imbalance Rate. To create varying levels of imbalance, each client is configured to contain only 2 out of 10 classes of data, with $high = 1$ and $low = \{0.2, 0.4, 0.6, 0.8, 1\}$. A greater disparity between these low and high values corresponds to an increased rate of imbalance. The data presented in Figure 5b reveal that pFedBASC consistently outperforms in terms of accuracy across varied imbalance rates, demonstrating its robustness to such imbalances.



(**a**) Non-iid (**b**) Imbalance

Figure 5. The accuracy of pFedBASC and baseline methods varies depending on (**a**) non-iid levels, with the *x*-axis displaying number of classes held by a single client, labeled as 'class', and (**b**) rates of imbalance, where the *x*-axis represents the value of the low rate used in the random function to generate imbalance rates, denoted by '*Low*', when $high = 1$, on CIFAR-10.

5.5. FedHype Scalability

The advantage of FedHype is its compatibility with existing FL aggregation algorithms, benefiting from their contributions. FedHype can not only be combined with our pFedBASC but also integrated with some of the baseline algorithms. The results are shown in Table 4. In this table, the accuracy improvement of the combined algorithm compared to the original one is presented. The results show that by incorporating FedHype, the efficiency of all baseline algorithms is significantly improved.

Table 4. The accuracy variation when FedHype is combined with other baseline algorithms.

Method	MNIST		FMNIST		CIFAR10		CIFAR100	
	non-iid	iid	non-iid	iid	non-iid	iid	non-iid	iid
FedHype + FedAvg	−1.22%	−0.9%	1.18%	1.13%	10.75%	12.33%	10.53%	10.94%
FedHype + FedProx	−1.09%	−0.66%	0.3%	0.82%	6.54%	13.65%	12.95%	10.06%
FedHype + FedDyn	−0.98%	−0.66%	−0.41%	0.98%	7.47%	13.16%	14.58%	9.39%
FedHype + FedBN	−1.03%	−1.55%	−1.07%	0.98%	0.13%	29.35%	3.67%	4.25%
FedHype + pFedLa	−0.46%	−0.25%	0.26%	2.51%	8.76%	16.18%	13.24%	15.62%
FedHype + FedFomo	4.74%	5.25%	7.8%	12.44%	13.83%	40.25%	16.7%	22.88%

6. Conclusions

This paper presents pFedBASC, a novel blockchain-assisted semi-centralized pFL framework designed to improve accuracy and efficiency in real-world environments, particularly suited to the distributed data landscapes of IoT. By integrating both centralized and decentralized FL frameworks, pFedBASC effectively leverages network connections within IoT to facilitate FL tasks while decentralizing aggregation operations across clients to minimize communication overhead. The framework not only involves meticulous system modeling and formal process descriptions but also includes the design of blockchain block structures—*download blocks*, *upload blocks*, and *evaluation blocks*—that guide the deployment of BFL systems. Our extensive experiments demonstrate pFedBASC's excellent accuracy, efficiency, and robustness to imbalance rates. Embedding FedHype into pFedBASC, which utilizes small local HNs for client-side knowledge exchange, addresses the challenges of pFL training and data heterogeneity by training personalized target networks through local HN aggregation. This innovative approach not only alleviates issues of data heterogeneity but also conserves communication costs. Moreover, HNs enable collaborative training among clients with heterogeneous models by generating parameters for TNs with varying architectures.

While our experiments have validated the effectiveness of the method, our next focus will be on providing theoretical proof of algorithm convergence for the semi-centralized framework. In pFedBASC, the initial design did not address FL tasks regarding blockchain consensus mechanisms effectively, and evaluation blocks, despite being designed, were not utilized. Future work will refine the integration of the blockchain algorithm design and FL. Additionally, since HNs can generate parameters for TNs with varying architectures, this supports collaborative training among clients with heterogeneous models. Moving forward, we plan to explore whether pFedBASC can maintain satisfactory model accuracy in heterogeneous model FL scenarios.

Author Contributions: Conceptualization and methodology, Y.Z. and H.X.; data curation, X.P.; writing—original draft preparation, Y.Z.; writing—review and editing, H.X. and X.P.; supervision, H.X. All authors have read and agreed to the published version of the manuscript.

Funding: This research is supported in part by the National Natural Science Foundation of China under Grant 62102212.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; y Arcas, B.A. Communication-Efficient Learning of Deep Networks from Decentralized Data. In Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, Fort Lauderdale, FL, USA, 20–22 April 2017; Volume 54, pp. 1273–1282. Available online: http://proceedings.mlr.press/v54/mcmahan17a.html (accessed on 10 September 2023).
2. Bonawitz, K.A.; Ivanov, V.; Kreuter, B.; Marcedone, A.; McMahan, H.B.; Patel, S.; Ramage, D.; Segal, A.; Seth, K. Practical Secure Aggregation for Privacy-Preserving Machine Learning. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, 30 October–3 November 2017; pp. 1175–1191. [CrossRef]
3. Guo, S.; Zhang, K.; Gong, B.; Chen, L.; Ren, Y.; Qi, F.; Qiu, X. Sandbox Computing: A Data Privacy Trusted Sharing Paradigm Via Blockchain and Federated Learning. *IEEE Trans. Comput.* **2023**, *72*, 800–810. [CrossRef]
4. Lu, Y.; Huang, X.; Zhang, K.; Maharjan, S.; Zhang, Y. Blockchain and Federated Learning for 5G Beyond. *IEEE Netw.* **2021**, *35*, 219–225. [CrossRef]
5. Feng, L.; Zhao, Y.; Guo, S.; Qiu, X.; Li, W.; Yu, P. BAFL: A Blockchain-Based Asynchronous Federated Learning Framework. *IEEE Trans. Computers* **2022**, *71*, 1092–1103. [CrossRef]
6. Gao, L.; Li, L.; Chen, Y.; Xu, C.; Xu, M. FGFL: A blockchain-based fair incentive governor for Federated Learning. *J. Parallel Distributed Comput.* **2022**, *163*, 283–299. [CrossRef]
7. Nguyen, D.C.; Hosseinalipour, S.; Love, D.J.; Pathirana, P.N.; Brinton, C.G. Latency Optimization for Blockchain-Empowered Federated Learning in Multi-Server Edge Computing. *IEEE J. Sel. Areas Commun.* **2022**, *40*, 3373–3390. [CrossRef]
8. Qu, Y.; Gao, L.; Xiang, Y.; Shen, S.; Yu, S. FedTwin: Blockchain-Enabled Adaptive Asynchronous Federated Learning for Digital Twin Networks. *IEEE Netw.* **2022**, *36*, 183–190. [CrossRef]
9. Wang, Y.; Peng, H.; Su, Z.; Luan, T.H.; Benslimane, A.; Wu, Y. A Platform-Free Proof of Federated Learning Consensus Mechanism for Sustainable Blockchains. *IEEE J. Sel. Areas Commun.* **2022**, *40*, 3305–3324. [CrossRef]
10. Wang, W.; Wang, Y.; Huang, Y.; Mu, C.; Sun, Z.; Tong, X.; Cai, Z. Privacy protection federated learning system based on blockchain and edge computing in mobile crowdsourcing. *Comput. Netw.* **2022**, *215*, 109206. [CrossRef]
11. Rückel, T.; Sedlmeir, J.; Hofmann, P. Fairness, integrity, and privacy in a scalable blockchain-based federated learning system. *Comput. Netw.* **2022**, *202*, 108621. [CrossRef]
12. Cui, L.; Su, X.; Zhou, Y. A Fast Blockchain-Based Federated Learning Framework With Compressed Communications. *IEEE J. Sel. Areas Commun.* **2022**, *40*, 3358–3372. [CrossRef]
13. Li, Z.; Yu, H.; Zhou, T.; Luo, L.; Fan, M.; Xu, Z.; Sun, G. Byzantine Resistant Secure Blockchained Federated Learning at the Edge. *IEEE Netw.* **2021**, *35*, 295–301. [CrossRef]
14. Pokhrel, S.R.; Choi, J. Federated Learning With Blockchain for Autonomous Vehicles: Analysis and Design Challenges. *IEEE Trans. Commun.* **2020**, *68*, 4734–4746. [CrossRef]
15. Li, Y.; Chen, C.; Liu, N.; Huang, H.; Zheng, Z.; Yan, Q. A Blockchain-Based Decentralized Federated Learning Framework with Committee Consensus. *IEEE Netw.* **2021**, *35*, 234–241. [CrossRef]
16. Feng, L.; Yang, Z.; Guo, S.; Qiu, X.; Li, W.; Yu, P. Two-Layered Blockchain Architecture for Federated Learning Over the Mobile Edge Network. *IEEE Netw.* **2022**, *36*, 45–51. [CrossRef]
17. Tan, A.Z.; Yu, H.; Cui, L.; Yang, Q. Towards Personalized Federated Learning. *IEEE Trans. Neural Networks Learn. Syst.* **2023**, *34*, 9587–9603. [CrossRef]
18. Mansour, Y.; Mohri, M.; Ro, J.; Suresh, A.T. Three Approaches for Personalization with Applications to Federated Learning. *arXiv* **2020**, arXiv:2002.10619.
19. Sun, J.; Chen, T.; Giannakis, G.B.; Yang, Z. Communication-Efficient Distributed Learning via Lazily Aggregated Quantized Gradients. In Proceedings of the Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, Vancouver, BC, Canada, 8–14 December 2019; pp. 3365–3375. Available online: https://proceedings.neurips.cc/paper/2019/hash/4e87337f366f72daa424dae11df0538c-Abstract.html (accessed on 25 November 2023).
20. Fallah, A.; Mokhtari, A.; Ozdaglar, A.E. Personalized Federated Learning with Theoretical Guarantees: A Model-Agnostic Meta-Learning Approach. In Proceedings of the Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, Virtual, 6–12 December 2020. Available online: https://proceedings.neurips.cc/paper/2020/hash/24389bfe4fe2eba8bf9aa9203a44cdad-Abstract.html (accessed on 27 November 2023).
21. Dinh, C.T.; Vu, T.T.; Tran, N.H.; Dao, M.N.; Zhang, H. FedU: A Unified Framework for Federated Multi-Task Learning with Laplacian Regularization. *arXiv* **2021**, arXiv:2102.07148.
22. Smith, V.; Chiang, C.; Sanjabi, M.; Talwalkar, A. Federated Multi-Task Learning. In Proceedings of the Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, Long Beach, CA, USA, 4–9 December 2017; pp. 4424–4434. Available online: https://proceedings.neurips.cc/paper/2017/hash/6211080fa89981f66b1a0c9d55c61d0f-Abstract.html (accessed on 27 November 2023).
23. Collins, L.; Hassani, H.; Mokhtari, A.; Shakkottai, S. Exploiting Shared Representations for Personalized Federated Learning. In Proceedings of the 38th International Conference on Machine Learning, ICML 2021, Virtual Event, 18–24 July 2021; Volume 139, pp. 2089–2099. Available online: http://proceedings.mlr.press/v139/collins21a.html (accessed on 28 November 2023).

24. Huang, Y.; Chu, L.; Zhou, Z.; Wang, L.; Liu, J.; Pei, J.; Zhang, Y. Personalized Cross-Silo Federated Learning on Non-IID Data. In Proceedings of the Proceedings of the AAAI Conference on Artificial Intelligence, Virtual Event, 2–9 February 2021; pp. 7865–7873. [CrossRef]
25. Ouyang, X.; Xie, Z.; Zhou, J.; Huang, J.; Xing, G. ClusterFL: A similarity-aware federated learning system for human activity recognition. In Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services, Virtual Event, WI, USA, 24 June–2 July 2021; pp. 54–66. [CrossRef]
26. Li, X.; Jiang, M.; Zhang, X.; Kamp, M.; Dou, Q. FedBN: Federated Learning on Non-IID Features via Local Batch Normalization. In Proceedings of the 9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, 3–7 May 2021; OpenReview.net. 2021. Available online: https://openreview.net/forum?id=6YEQUn0QICG (accessed on 15 February 2024).
27. Arivazhagan, M.G.; Aggarwal, V.; Singh, A.K.; Choudhary, S. Federated Learning with Personalization Layers. *arXiv* **2019**, arXiv:1912.00818.
28. Oh, J.; Kim, S.; Yun, S. FedBABU: Towards Enhanced Representation for Federated Image Classification. *arXiv* **2021**, arXiv:2106.06042.
29. Zhang, M.; Sapra, K.; Fidler, S.; Yeung, S.; Álvarez, J.M. Personalized Federated Learning with First Order Model Optimization. In Proceedings of the 9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, 3–7 May 2021; OpenReview.net. 2021. Available online: https://openreview.net/forum?id=ehJqJQk9cw (accessed on 16 February 2024).
30. Schmidhuber, J. Learning to Control Fast-Weight Memories: An Alternative to Dynamic Recurrent Networks. *Neural Comput.* **1992**, *4*, 131–139. [CrossRef]
31. Shamsian, A.; Navon, A.; Fetaya, E.; Chechik, G. Personalized Federated Learning using Hypernetworks. In Proceedings of the 38th International Conference on Machine Learning, ICML 2021, Virtual Event, 18–24 July 2021; Volume 139, pp. 9489–9502. Available online: http://proceedings.mlr.press/v139/shamsian21a.html (accessed on 15 February 2024).
32. Ma, X.; Zhang, J.; Guo, S.; Xu, W. Layer-wised Model Aggregation for Personalized Federated Learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, 18–24 June 2022; pp. 10082–10091. [CrossRef]
33. Chen, X.; Huang, Y.; Xie, Z.; Pang, J. HyperFedNet: Communication-Efficient Personalized Federated Learning Via Hypernetwork. *arXiv* **2024**, arXiv:2402.18445. https://doi.org/10.48550/arXiv.2402.18445.
34. Chen, H.; Chao, W. On Bridging Generic and Personalized Federated Learning for Image Classification. In Proceedings of the Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, 25–29 April 2022; OpenReview.net. 2024. Available online: https://openreview.net/forum?id=I1hQbx10Kxn (accessed on 18 February 2024).
35. Ha, D.; Dai, A.M.; Le, Q.V. HyperNetworks. In Proceedings of the 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, 24–26 April 2017. Available online: https://openreview.net/forum?id=rkpACe1lx (accessed on 30 November 2023).
36. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [CrossRef]
37. Xiao, H.; Rasul, K.; Vollgraf, R. Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv* **2017**, arXiv:1708.07747.
38. Krizhevsky, A.; Hinton, G. Learning Multiple Layers of Features from Tiny Images. In *Handbook of Systemic Autoimmune Diseases*. 2009. Available online: https://www.cs.utoronto.ca/~kriz/learning-features-2009-TR.pdf (accessed on 15 February 2024).
39. Li, T.; Sahu, A.K.; Zaheer, M.; Sanjabi, M.; Talwalkar, A.; Smith, V. Federated Optimization in Heterogeneous Networks. In Proceedings of the Machine Learning and Systems 2020, MLSys 2020, Austin, TX, USA, 2–4 March 2020; mlsys.org. 2020. Available online: https://proceedings.mlsys.org/paper_files/paper/2020/file/1f5fe83998a09396ebe6477d9475ba0c-Paper.pdf (accessed on 15 February 2024).
40. Zhu, Z.; Hong, J.; Zhou, J. Data-Free Knowledge Distillation for Heterogeneous Federated Learning. In Proceedings of the 38th International Conference on Machine Learning, ICML 2021, Virtual Event, 18–24 July 2021; Volume 139, pp. 12878–12889. Available online: http://proceedings.mlr.press/v139/zhu21b.html (accessed on 15 February 2024).
41. Acar, D.A.E.; Zhao, Y.; Navarro, R.M.; Mattina, M.; Whatmough, P.N.; Saligrama, V. Federated Learning Based on Dynamic Regularization. In Proceedings of the 9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, 3–7 May 2021; OpenReview.net. 2021. Available online: https://openreview.net/forum?id=B7v4QMR6Z9w (accessed on 15 February 2024).
42. Tan, J.; Zhou, Y.; Liu, G.; Wang, J.H.; Yu, S. pFedSim: Similarity-Aware Model Aggregation Towards Personalized Federated Learning. *arXiv* **2023**, arXiv:2305.15706. https://doi.org/10.48550/arXiv.2305.15706.