Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

A Novel Approach for Conducting Asynchronous Federated Learning with Blockchain Integration

Sung-Jung Hsiao

sung.jung@gs.takming.edu.tw

Takming University of Science and Technology

Research Article

Keywords: Blockchain, asynchronous portrait, federated learning, linear regression, variance

Posted Date: September 9th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-10963813/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

A Novel Approach for Conducting Asynchronous Federated Learning with Blockchain Integration

Sung-Jung Hsiao *

Department of Information Technology, Takming University of Science and Technology, Taipei City, 11451, Taiwan

*Corresponding Author: Sung-Jung Hsiao. Email: sungjung@gs.takming.edu.tw

Received: XX Month 202X; Accepted: XX Month 202X

Abstract: Conducting privacy-preserving collaborative learning over vertically partitioned data has emerged as a promising solution for multiparty joint modeling tasks. In such settings, data holders—including government agencies, financial institutions, and e-commerce platforms—can collaboratively train models without sharing their raw data or relying on trusted third parties. However, most existing vertical federated learning (VFL) approaches are constrained to synchronous training processes, which limits the efficiency of systems under heterogeneous computing and communication conditions. To address this limitation, an asynchronous VFL framework that improves the efficiency of training while preserving data privacy is proposed in this paper. To further provide enhanced security, trust, and participant engagement, blockchain technology is integrated into the proposed framework. Specifically, blockchain consensus mechanisms and smart contracts are leveraged to provide decentralized coordination, transparent model evaluation, and incentive-driven collaboration capabilities. Unlike traditional VFL systems that rely on a centralized coordinator, the proposed approach replaces the central server with a blockchain network. Task rewards are encapsulated within smart contracts, enabling the automatic and fair distribution of incentives based on a model quality evaluation. Participants are rewarded according to their contributions to the training process, thereby encouraging active and reliable participation. Furthermore, the existing blockchain-based federated learning strategies are reviewed and incorporated herein from the perspectives of model quality assessments, weight aggregation processes, and incentive mechanisms. The proposed framework has the potential to provide improved scalability, robustness, and participation efficiency in practical multiparty learning scenarios.

Keywords: Blockchain, asynchronous portrait, federated learning, linear regression, variance

1. Introduction

As an emerging solution for multiparty collaborative modeling tasks, privacy-preserving learning based on vertically partitioned data has demonstrated considerable potential. In this paradigm, data holders—such as government agencies, privacy-sensitive financial institutions, and e-commerce companies—collaborate throughout the entire model-building process without relying on a trusted third party to store or aggregate data. However, the existing vertical federated learning (VFL) algorithms remain largely constrained to synchronous computations, which limits the efficiency of systems when participating parties possess heterogeneous computational or communication resources.

To enhance the scalability and practical performance of VFL under such resource-imbalanced conditions, it is essential to design asynchronous VFL algorithms that preserve data privacy. This study further incorporates blockchain technology to provide a tamper-resistant, auditable, and decentralized coordination mechanism, thereby strengthening the privacy and trustworthiness of the constructed federated learning system [1].

Blockchain technologies, particularly consensus mechanisms and smart contracts, provide intrinsic incentive structures that can effectively motivate active participation among multiple stakeholders. Leveraging these properties, the integration of blockchain with federated learning has emerged as a promising direction for addressing the persistent challenges encountered in collaborative model training scenarios. In this study, several blockchain-based federated learning algorithms are examined from multiple incentive perspectives, including model quality assessments, weight allocation schemes, and reward distributions.

From a system deployment perspective, the most significant architectural modification introduced by blockchain-enabled federated learning is the replacement of the traditional centralized coordinator with a decentralized blockchain network. The other components of the federated learning workflow remain largely consistent with the

conventional designs. An overview of the deployment architecture is presented in Figure 1. As illustrated, four participants, three enterprises and one regulatory authority jointly constitute the federated learning platform. Each enterprise connects to a private blockchain peer node through its business gateway, and all data exchanges and communications within the platform are conducted exclusively through the blockchain network.

Within each enterprise, a server receives tasks forwarded by the business gateway and delegates them to local workers for machine learning computations. The resulting model updates are subsequently aggregated back into the blockchain network along the reverse communication path. This iterative process continues until the learning task converges. For the regulatory participant, a dedicated regulatory gateway enables the system activities to be monitored in real time, allowing prompt interventions when abnormal or non-compliant operations occur [2][3].

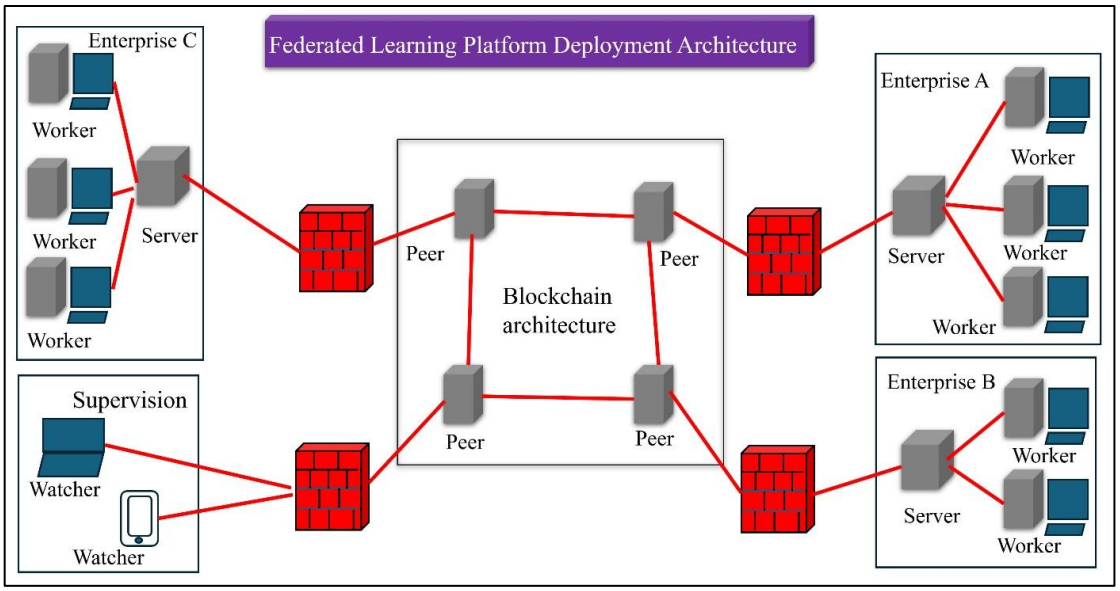


Figure 1. Deployment architecture of the federated learning platform.

2. Asynchronous VFL algorithm

Within the VFL setting, a suite of fully asynchronous stochastic optimization algorithms is introduced in this study; these algorithms include completely asynchronous federated stochastic gradient descent on reliable vertically partitioned data (CAFSGD-RVPD) and two corresponding variance-reduced extensions, namely,

CAFSVRG-RVPD and CAFSAGA-RVPD. For all three algorithms, we establish rigorous convergence guarantees under strong convexity conditions and demonstrate that convergence is achieved without imposing any assumptions on bounded communication delays—an important property for practical deployments in heterogeneous environments. Furthermore, we provide a detailed technical analysis of the model privacy, data privacy, and communication overhead associated with the proposed methods.

Comprehensive empirical evaluations conducted on multiple vertically partitioned datasets corroborate the theoretical findings and show that the asynchronous algorithms consistently outperform their synchronous counterparts in terms of computational efficiency and convergence speed. The asynchronous update mechanism enables higher throughput and improved scalability while preserving provable convergence rates under strong convexity conditions. The algorithmic framework encompasses a baseline CSGD-based method and two variance-reduced variants that are separately derived from CSVRG and CSAGA, all of which are robust against both exact and approximate inference attacks.

In practical federated learning systems, malicious participants may intentionally submit corrupted gradient updates to compromise the integrity of models, whereas free-riding or inactive nodes may contribute insufficient information, resulting in biased or degraded model performance. Ensuring sustained, fair, and trustworthy participation among multiple parties thus remains a critical challenge for industrial-scale federated learning deployments [4][5].

From the algorithmic design perspective, a substantial body of research has been devoted to implementing learning over vertically partitioned data. Numerous privacy-preserving federated learning algorithms have been developed for various applications, including collaborative statistical analyses, linear regression, association rule mining, logistic regression, XGBoost, random forests, and support vector machines. However, the existing VFL approaches predominantly rely on synchronous computations. Such synchronous protocols often suffer from low efficiency levels, as they fail to utilize idle computational resources and are constrained by the slowest participating party.

Stochastic gradient descent (SGD) and its variance-reduced variants have become the dominant optimization methods for large-scale machine learning cases. In particular, each iteration of CSGD independently samples a data point and updates the corresponding model using its stochastic gradient. While this stochasticity yields a low per-iteration computational cost, it also introduces high gradient variance. To mitigate this issue, our approach incorporates several variance reduction techniques for CSGD,

including CSVRG, CSAG, and CSAGA. Despite these advances, a notable gap remains in the literature: few studies have addressed how to train vertically partitioned models in a fully asynchronous manner while simultaneously ensuring both data privacy and model privacy.

To address this challenge, we propose an asynchronous VFL framework consisting of the CAFSGD-RVPD algorithm and its two variance-reduced variants based on CSVRG and CSAGA. More importantly, we establish theoretical convergence rates for all three algorithms under strong convexity, demonstrating their robustness and efficiency in asynchronous environments [6][7].

3. Related work and literature review

This study provides a concise review of the existing asynchronously distributed stochastic optimization algorithms and CSGD-type methods that are relevant to learning over vertically partitioned data. A substantial body of work has investigated asynchronously distributed stochastic optimization schemes for large-scale machine learning tasks; however, most existing approaches were designed for horizontal federated learning rather than vertical settings.

For achieving smooth convex optimization, Zhao et al. proposed an asynchronous stochastic algorithm based on CSVRG and established its linear convergence rate. Mania et al. introduced the perturbed iteration framework to analyze asynchronous CSVRG algorithms that rely on sparse gradient updates. Huo et al. extended asynchronous CSVRG to nonconvex optimization settings and proved its sublinear convergence guarantees. Leblond et al. developed an asynchronous CSAGA algorithm and demonstrated its linear convergence properties.

With respect to nonsmooth regularized convex optimization, Gu et al. independently proposed an asynchronous stochastic proximal gradient method based on CSVRG and proved its linear convergence. In the context of vertically partitioned data, Zhang et al. introduced FD-CSVRG, the first privacy-preserving CSGD-based method tailored to VFL; however, their algorithm operates under a synchronous computation model.

To our knowledge, FDML remains the only asynchronous algorithm that has been designed specifically for VFL scenarios. However, FDML strongly depends on the assumption that communication delays are bounded by a known upper limit. In practical distributed systems, such delay bounds are typically unknown or difficult to accurately estimate. As a result, the convergence of FDML cannot be guaranteed in real-world deployments, leading to an inevitable loss of model expressiveness because of the mismatch between the theoretical delay assumptions and actual system behaviors. The

experimental results presented later in this study further validate this issue by comparing the classification accuracy values attained under realistic conditions.

In contrast, the algorithms proposed in this work are lossless in the sense that they impose no constraints or assumptions on the communication delays in either the theoretical analysis or the practical implementation. The key distinctions between our approach and FDML are summarized as follows [8].

(1) Although both the FDML algorithm and the method proposed in this study adopt asynchronously distributed computations, they differ fundamentally in terms of their system architectures. FDML is based on a master–worker framework, whereas the method introduced in this work is fully decentralized. A decentralized architecture offers greater flexibility and fault tolerance; in fact, a master–worker structure can be regarded as a static and constrained special case of a decentralized system. More importantly, this study not only proposes a decentralized computational method but also provides a theoretical analysis of its convergence rate, which represents a key contribution beyond the FDML process.

(2) The FDML algorithm supports a decreasing step size, whereas the method proposed in this study allows the use of a fixed step size. This makes the proposed algorithm more suitable for asynchronously distributed settings, as it eliminates the need to adjust the step size at every iteration [9] [10].

(3) The FDML algorithm supports only CSGD, whereas the method proposed in this study extends beyond CSGD and additionally includes variants based on CSVRG and CSAGA.

As mentioned earlier, CSGD-type algorithms have become popular approaches for solving large-scale machine learning problems. Our work begins by briefly reviewing the update framework of CSGD-type algorithms, including multivariate variance reduction techniques. Specifically, given an unbiased stochastic gradient ν satisfying $\mathbb{E}_{\upsilon} = \nabla f(\omega)$, the update rule of a CSGD-type algorithm can be expressed as shown in Equation (1):

$$\omega \leftarrow \omega - \zeta\nu \tag{1}$$

where ζ denotes the learning rate. Our study introduces the specific forms of the stochastic gradients used in the CSGD, CSVRG, and CSAGA algorithms. In the CSGD algorithm, each iteration involves independently sampling a data point (x_i, y_i) and updating the model using the corresponding stochastic gradient $\nabla f_i(\omega)$, as shown in Equation (2):

$$v = \nabla f_i(\omega) \tag{2}$$

The CSVRG algorithm does not directly use the stochastic gradient $\nabla f_i(\omega)$. Instead, it performs updates using an unbiased stochastic gradient v, which is defined as shown in Equation (3).

$$v = \nabla f_i(\omega) - \nabla f_i(\widetilde{\omega}) + \nabla f(\widetilde{\omega}) \tag{3}$$

where $\widetilde{\omega}$ denotes the value of ω at a certain reference iteration within an epoch. The unbiased stochastic gradient v used in the CSAGA algorithm is given as presented in Equation (4).

$$v = \nabla f_i(\omega) - \alpha_i + \frac{1}{l}\sum_{i=1}^{l} \alpha_i \tag{4}$$

where α_i denotes the most recent historical gradient of $\nabla f_i(\omega)$.

4. Problem formulation

In this study, our method is considered a model of the form $\omega^T x$. Suppose we are given a training set $S = \{(x_i, y_i)\}_{i=1}^{l}$, where $x_i \in \mathbb{R}^d$. For binary classification problems, the labels satisfy $y_i \in \{+1, -1\}$, whereas for regression problems, $y_i \in \mathbb{R}$. The loss associated with a sample (x_i, y_i) and a model parameter ω is represented by a general loss function $L(\omega^T x_i, y_i)$. Accordingly, the computational method developed in this work is aimed at optimizing the following empirical risk minimization problem, as stated in Equation (5):

$$\min_{\omega \in \mathbb{R}^d} f(\omega) = \frac{1}{l}\sum_{i=1}^{l} L(\omega^T x_i, y_i) + g(\omega) \tag{5}$$

where $g(\omega)$ denotes a regularization term, and each function $f_i: \mathbb{R}^d$ is smooth and may be nonconvex. Clearly, the empirical risk minimization problem is a special case of this general formulation. Beyond empirical risk minimization, Equation (5) also encompasses a broad class of regularized learning problems, such as regularized logistic regression, ridge regression, and least-squares support vector machines. As previously discussed, in many real-world machine learning applications, the input training sample (x, y) is vertically partitioned among q participating parties, which are denoted by $\{\mathcal{G}_1, \cdots, \mathcal{G}_q\}$. Consequently, the feature vector can be written as $x = [x_{\mathcal{G}_1}, x_{\mathcal{G}_2}, \cdots, x_{\mathcal{G}_q}]$, where $x_{\mathcal{G}_\ell} \in \mathbb{R}^{d_\ell}$ is stored by the ℓ-th party, and the dimensions satisfy

$\sum_{\ell=1}^{q} d_\ell = d$. Based on whether a party possesses label information, our framework categorizes participants into active and passive parties. The active party possesses both feature data and label information, whereas the passive parties hold only feature data. Let D^ℓ denote the data stored at the ℓ-th party, and assume that the labels y_i are available exclusively at the active party. Under this setting, the algorithmic objective can be summarized as enabling the active and passive parties to collaboratively solve a regularized empirical risk minimization problem over the vertically partitioned dataset $\{D^\ell\}_{\ell=1}^{q}$. This is achieved by employing asynchronous parallel CSGD and its variants, namely, CSVG and CSAGA, while simultaneously ensuring data privacy [11] [12].

5. Proposed computational approach for this study

The originality and significance of this study are highlighted in the following section. We first present the overall framework of an asynchronous VFL algorithm. Building upon this framework, we propose three asynchronous VFL algorithms based on the CSGD family, namely, CAFSGD-RVPD, CAFSVRG-RVPD, and CAFSAGA-RVPD.

5.1 Algorithmic framework

As described earlier, CAFSGD-RVPD, CAFSVRG-RVPD, and CAFSAGA-RVPD are privacy-preserving asynchronous federated learning algorithms. Their system architecture is illustrated in Figure 2. In the following sections, we provide detailed descriptions of the tree-structured communication mechanism, data privacy scheme, and model privacy scheme [13][14][15].

To compute $\omega^T x_i$, our method requires the corresponding partial results to be collected from multiple participating parties. Zhang et al. (2018) introduced an efficient tree-structured communication scheme for obtaining global summations, which significantly outperformed the naive approach of transmitting all local results to a central coordinator for aggregation purposes. The efficiency of an aggregation strategy depends on both the communication and computational costs incurred. In simple aggregation schemes, the master node bears the full burden of communication and summation. In contrast, tree-structured communication distributes these workloads evenly across the participating nodes, enabling substantially higher efficiency to be achieved when numerous aggregation operations are invoked concurrently.

In this study, we adopt the tree-structured communication scheme to compute $\omega^T x_i$. Notably, the scheme is implemented asynchronously, meaning that the iteration counters of $\omega_{\mathcal{G}_\ell}$ across different parties are not synchronized. This stands in sharp contrast with the synchronous approaches that were commonly employed in prior work, where all $\omega_{\mathcal{G}_\ell}$ values were required to remain in the same iteration round.

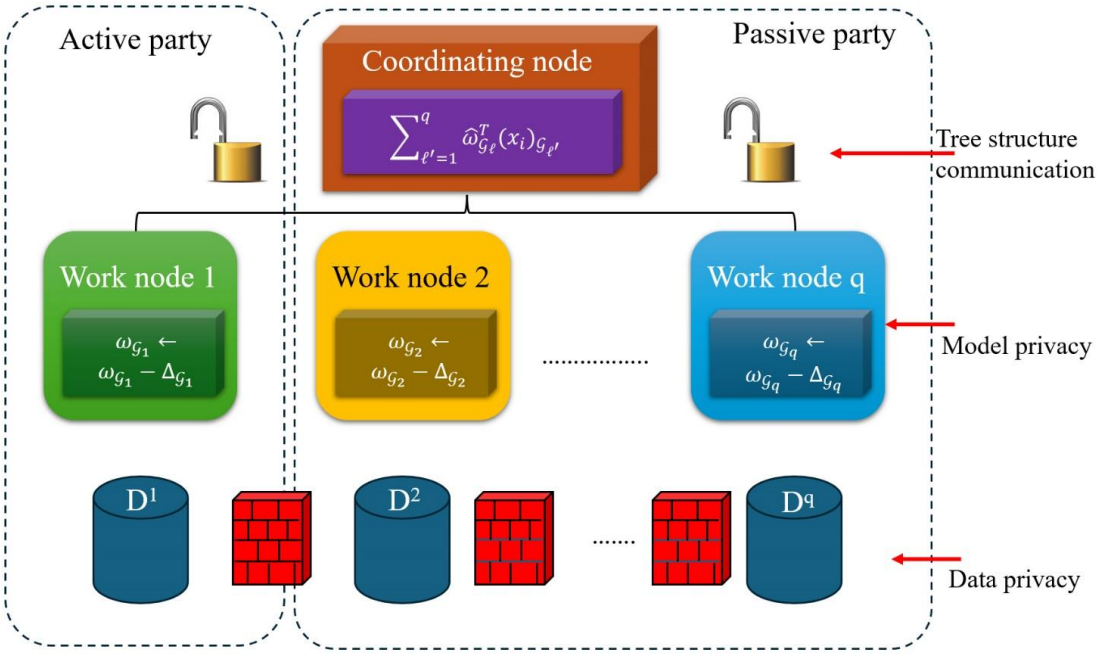


Figure 2. System architecture of privacy-preserving asynchronous federated learning algorithms.

5.2 Scenario case study

To illustrate how the proposed algorithms can be applied in a practical setting, we consider a feasible scenario in which three companies collaboratively train a machine learning model under privacy-preserving constraints using federated learning. The objective of this case study is to solve a real-world prediction task based on the UCI Credit Card dataset, where the goal is to predict whether a user will default on a loan given multiple personal and financial attributes.

The dataset contains 23 features for each user, including their credit limit, gender, education level, marital status, age, repayment status, monthly bill amounts, and monthly payment amounts. The corresponding label indicates whether the user defaulted on their loan (1 for the default case and 0 for the nondefault case). In total, the dataset consists of 30,000 samples.

Based on this dataset, we construct the following federated learning scenario. Suppose that Companies A, B, and C each hold different user feature subsets:

- Company A has 9 features, including the credit limit, gender, age, and six months of repayment statuses.
- Company B has 9 features, including the credit limit, gender, age, education level, and six months of monthly bill amounts.
- Company C has 9 features, including gender, age, marital status, and six months of monthly payment amounts.

Importantly, the three companies do not share identical customer bases; that is, their sample IDs are not perfectly aligned. This setting reflects a realistic VFL environment in which different organizations hold heterogeneous feature spaces and partially overlapping user populations. The proposed algorithms—CAFSGD-RVPD, CAFSVRG-RVPD, and CAFSAGA-RVPD—are designed to operate effectively under such conditions, enabling a privacy-preserving collaborative model training procedure without exposing raw user data.

First, we assume that the data of each company have been preprocessed into the standard data layout used for VFL, which is consistent with the UCICreditCard dataset. Specifically, Company A provides nine features, including the credit limit, gender, age, and monthly repayment status over the past six months, i.e., $x_{\mathcal{G}_1} = [x_1,\ x_2,\ x_6, \cdots,\ x_{11}] \in \mathbb{R}^9$. Company B provides seven features, including the education level and the total monthly bill amounts over the past six months, i.e., $x_{\mathcal{G}_2} = [x_3,\ x_{12},\ \cdots,\ x_{17}] \in \mathbb{R}^7$. Company C provides seven features, including marital statuses and the total monthly bill amounts over the past six months, i.e., $x_{\mathcal{G}_3} = [x_4,\ x_{18},\ \cdots,\ x_{23}] \in \mathbb{R}^7$. All three companies share the same set of 30,000 common sample IDs. Among them, 24,000 samples are used for training, and the remaining 6,000 samples are used for testing. In this scenario, all the participating companies are loan-issuing institutions, and labels are available at Companies A, B, and C. We adopt an ℓ_2-regularized logistic regression model for joint modeling purposes; therefore, the learning problem implemented under this setting can be formulated as shown in Equation (6):

$$\min_{\omega \in \mathbb{R}^d} f(\omega) \coloneqq \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}(\theta_i,\ y_i) + \lambda \sum_{m=1}^{q} \left\| \omega_{\mathcal{G}_m} \right\|^2 \tag{6}$$

where $q = 3$ denotes the number of participating companies (parties), $n = 24{,}000$ is the number of training samples, and $d = 23$ is the total feature dimensionality. For the

i-th sample, we define the linear predictor $\theta = \omega^T x_i = \sum_{m=1}^{3} \omega_{\mathcal{G}_m}^T (x_i)_{\mathcal{G}_m}$, where the vertically partitioned feature vector and model parameter vector are concatenated as $x_i = [(x_i)_{\mathcal{G}_1},\ (x_i)_{\mathcal{G}_2},\ (x_i)_{\mathcal{G}_3}]$ and $\omega = [\omega_{\mathcal{G}_1},\ \omega_{\mathcal{G}_2},\ \omega_{\mathcal{G}_3}]$, respectively. Moreover, the feature blocks $(x_i)_{\mathcal{G}_m}$ and the corresponding parameter blocks $\omega_{\mathcal{G}_m}$ are stored locally at their respective companies; i.e., each company maintains only its own features and model parameters.

6. Theoretical analysis and discussion

In this study, we conduct a theoretical analysis of the convergence, security, and computational complexity of CAFSGD-RVPD, CAFSVRG-RVPD, and CAFSAGA-RVPD.

6.1 Convergence analysis

First, we introduce several basic assumptions and then present the convergence results obtained for CAFSGD-RVPD, CAFSVRG-RVPD, and CAFSAGA-RVPD. Specifically, we begin with the strong convexity assumption, followed by the Lipschitz smoothness assumption and the boundedness of block-coordinate gradients. These assumptions are standard and widely adopted in convex analyses.

Assumption (Strong Convexity). For the strongly convex case, the differentiable functions f_i in Eq. (5) ($i \in \{1, \ldots, l\}$) are μ strongly convex for some $\mu > 0$; that is, $\forall\omega, \forall\omega'$, this method yields Equation (7):

$$f_i(\omega) \geq f_i(\omega') + \langle \nabla f_i(\omega'),\ \omega - \omega' \rangle + \frac{\mu}{2} \|\omega - \omega'\|^2 \tag{7}$$

Moreover, under the Lipschitz smoothness assumption, each function f_i in Equation (5) for $i \in \{1, \cdots,\ l\}$ is L-smooth with a constant L. That is, for all ω and ω', Equation (8) holds:

$$\|\nabla f_i(\omega) - \nabla f_i(\omega')\| \leq L\|\omega - \omega'\| \tag{8}$$

Each function f_i in Equation (5) $\forall i \in \{1, \cdots,\ l\}$ is also block-coordinate Lipschitz smooth. Specifically, for the ℓ-th block $\mathcal{G}_\ell$, there exists a constant L_ℓ such that for all ω and for all $\ell \in \{1, \cdots,\ q\}$, Equation (9) holds:

$$\left\|\nabla_{\mathcal{G}_\ell} f_i(\omega + \boldsymbol{U}_\ell \Delta_\ell) - \nabla_{\mathcal{G}_\ell} f_i(\omega)\right\| \leq L_\ell \|\Delta_\ell\| \tag{9}$$

where $\Delta_\ell \in \mathbb{R}^{d_\ell}$, $\boldsymbol{U}_\ell \in \mathbb{R}^{d \times d_\ell}$, and $\left[\boldsymbol{U}_1, \boldsymbol{U}_2, \cdots, \boldsymbol{U}_q\right] = \boldsymbol{I}_d$. On the basis of these assumptions, we define $L_{max} = max_{\ell=1,\cdots,q} L_\ell$.

We assume that the block-coordinate gradients are bounded. For each function $f_i(x)$ in Equation (5) with $i \in \{1, \cdots, l\}$, the block-coordinate gradient $\nabla_{\mathcal{G}_\ell} f_i(\omega)$ is bounded if a constant G exists such that $\left\|\nabla_{\mathcal{G}_\ell} f_i(\omega)\right\|^2 \leq G$ for all $i \in \{1, \cdots, l\}$, and $\ell \in \{1, \cdots, q\}$. Our subsequent methodological development process addresses the principal challenges that arise in the asynchronous analysis of the algorithm.

To present a convergence rate analysis of CAFSGD-RVPD, CAFSVRG-RVPD, and CAFSAGA-RVPD, it is necessary to specify the global update rules of these algorithms. Because the additive operation in $\omega_{\mathcal{G}_\ell} \leftarrow \omega_{\mathcal{G}_\ell} - \gamma \cdot \hat{v}^\ell$ is commutative, the order in which these updates are applied by the corresponding participants is immaterial. The global update rules for CAFSGD-RVPD, CAFSVRG-RVPD, and CAFSAGA-RVPD are as shown in Equation (10):

$$\omega_{t+1} = \omega_t - \gamma \boldsymbol{U}_{\psi(t)} \hat{v}_t^{\psi(t)} \tag{10}$$

Equation (10) defines the relationship between adjacent iterations; owing to the commutativity of the additive operation, this relationship does not conflict with the global iteration-tagging rule.

As noted above, the CAFSGD-RVPD, CAFSVRG-RVPD, and CAFSAGA-RVPD methods employ an asynchronous, tree-structured communication protocol to obtain $\widehat{\omega}^T x_i = \sum_{\ell'=1}^{q} (\widehat{\omega})_{\mathcal{G}_{\ell'}}^T (x_i)_{\mathcal{G}_{\ell'}}$, where $\widehat{\omega}$ denotes the (potentially) inconsistent copy of ω aggregated from different participants. For $\ell' \neq \ell$, the vector $(\widehat{\omega}_t)_{\mathcal{G}_{\ell'}}$ may be inconsistent with $(\omega_t)_{\mathcal{G}_{\ell'}}$. This means that some blocks of $\widehat{\omega}_t$ coincide with the corresponding blocks of ω_t (for example, $(\omega)_{\mathcal{G}_\ell} = (\widehat{\omega})_{\mathcal{G}_\ell}$, whereas other blocks may differ. To address this issue, we assume that the delays of the updates are uniformly bounded by a constant. We define an iteration set $D(t)$ such that the discrepancy between $\widehat{\omega}_t$ and ω_t can be written as shown in Equation (11):

$$\widehat{\omega}_t - \omega_t = \gamma \sum_{u \in D(t)} U_{\psi(u)} \hat{v}_u^{\psi(u)} \tag{11}$$

where for all $u \in D(t)$, we have that $u < t$. Assume that there exists a uniform upper bound τ imposed on the update delays, i.e., $\tau \geq t - \min\{t' | t' \in D(t)\}$.

Under the bounded overlap condition, for any iteration t in CAFSGD-RVPD, CAFSVRG-RVPD, and CAFSAGA-RVPD, there exists an upper bound τ such that $\tau \geq t - \min\{u | u \in D(t)\}$. Moreover, we assume that the cardinality of $\psi^{-1}(\ell,\ D(t))$ is uniformly bounded: there exists a constant η_2 such that for all t and all $\ell \in \{1, \cdots,\ q\}$, the condition holds.

Alternatively, we assume that the cardinality of $\psi^{-1}(\ell,\ D(t)$ is uniformly bounded. That is, for all t and for all $\ell \in \{1, \cdots,\ q\}$, the condition holds. For any block label ℓ and any set K, the number of elements in $\psi(\psi^{-1}(\ell,\ K))$ is bounded from above by η_2.

To obtain a solution with an accuracy of ϵ for Equation (5) using the CAFSGD-RVPD method, i.e., $f(\omega_t) - f(\omega^*) \leq \epsilon$, this approach needs to satisfy Equation (12):

$$\gamma = \frac{-L_{max} + \sqrt{L_{max}^2 + \frac{2\mu\epsilon(L^2 q\eta_1^2 + \eta_2 L^2 \tau)}{G\eta_1 q}}}{2L^2(q\eta_1^2 + \eta_2\tau)} \tag{12}$$

and the number of rounds υ(t) should satisfy the following conditions, as shown in Equation (13):

$$\upsilon(t) \geq \frac{2}{\mu} \frac{2L^2(q\eta_1^2 + \eta_2\tau)}{-L_{max} + \sqrt{L_{max}^2 + \frac{2\mu\epsilon(L^2 q\eta_1^2 + \eta_2 L^2 \tau)}{G\eta_1 q}}} \cdot \log\left(\frac{2(f(\omega_0) - f(\omega^*))}{\epsilon}\right) \tag{13}$$

The above equation shows that the convergence speed of the computational CAFSGD-RVPD method is $O(\frac{1}{\sqrt{\epsilon}}\log(\frac{1}{\epsilon}))$. This finding also indicates that if a smaller step size is desired for obtaining a more accurate solution, the convergence speed decreases. To obtain a solution with an accuracy of ϵ for Equation (5) using the computational CAFSGD-RVPD method, i.e., $\mathbb{E}f(\omega_t) - f(\omega^*) \leq \epsilon$, let $C = (\eta_1\gamma L^2 q\eta_1 + L_{max})\frac{\gamma^2}{2}$ and $= \frac{\gamma\mu}{2} - \frac{16L^2\eta_1 qC}{\mu}$, and choose a γ such that Equations (14-16) hold:

$$\rho < 0 \tag{14}$$

$$\frac{8L^2\eta_1 qC}{\rho\mu} \leq 0.5 \tag{15}$$

$$\gamma^3\left(\left(\frac{1}{2}+\frac{2C}{\gamma}\right)\eta_2\tau + 4\frac{C}{\gamma}\eta_1^2 q\right)\frac{\eta_1 qL^2G}{\rho} \leq \frac{\epsilon}{8} \tag{16}$$

Then, the number of inside rounds should satisfy $v(t) \geq \frac{log0.25}{\log(1-\rho)}$, and the number of outside rounds should satisfy $S \geq \frac{log\frac{2(f(\omega_0)-f(\omega^*))}{\epsilon}}{log\frac{4}{3}}$.

The above formula shows that the convergence speed of the CAFSVRG-RVPD algorithm is $O(\log(\frac{1}{\epsilon}))$. The convergence result of CAFSAGA-RVPD is obtained by using the CAFSGD-RVPD calculation method to obtain the solution for Equation (5) with an accuracy of ϵ; that is, $\mathbb{E}f(\omega_t) - f(\omega^*) \leq \epsilon$. Letting $C_0 = (\left(\frac{\eta_2}{2} + 3(\gamma q\eta_1^2 + L_{max})(\eta_1 + 2\eta_2)\right)\tau + (\gamma L^2 q\eta_1^2 + 8L_{max})\eta_1 q\eta_1)\gamma^4 L^2\eta_1 qG$, $c_1 = (\gamma L^2 q\eta_1^2 + L_{max})\gamma^2\eta_1 q2L^2$, $c_2 = 4(\gamma L^2 q\eta_1^2 + L_{max})\frac{L^2\eta_1^2 q}{l}\gamma^2$ and $\rho \in (1-\frac{1}{l},\ 1)$, we can choose an appropriate γ such that Equations (17-20) hold:

$$\frac{4c_0}{\gamma\mu(1-\rho)(\frac{\gamma\mu^2}{4}-2c_1-c_2)} \leq \frac{\epsilon}{2} \tag{17}$$

$$0 < 1 - \frac{\gamma\mu}{4} < 1 \tag{18}$$

$$-\frac{\gamma\mu^2}{4} + 2c_1 + c_2\left(1 + \frac{1}{1-\frac{1-\frac{1}{l}}{\rho}}\right) \leq 0 \tag{19}$$

$$-\frac{\gamma\mu^2}{4} + c_2 + c_1\left(2 + \frac{1}{1-\frac{1-\frac{1}{l}}{\rho}}\right) \leq 0 \tag{20}$$

The number of rounds should satisfy $v(t) \geq \frac{log\frac{2\left(2\rho-1+\frac{\gamma\mu}{4}\right)\left(f(\omega_0)-f(\omega^*)\right)}{\epsilon\left(\rho-1+\frac{\gamma\mu}{4}\right)\left(\frac{\gamma\mu^2}{4}-2c_1-c_2\right)}}{log\frac{1}{\rho}}$. The above formula shows that the convergence rate of the CAFSAGA-RVPD calculation method is $O(\log(\frac{1}{\epsilon}))$.

6.2 Security analysis

The data and model security of CAFSG-RVPD, CAFSVRG-RVPD, and CAFSAGA-RVPD are examined in this study under the semi-honest assumption. This assumption has been commonly adopted in prior work. Under semi-honest security, all participants follow the set protocol and perform computations correctly; however, they may exploit locally retained intermediate computation results to infer the data and models of other parties. Before providing a detailed discussion concerning data and model privacy, we first introduce the definitions of exact inference attacks and approximate inference attacks.

An exact inference attack on participant ℓ is defined as the adversary's ability to recover, exactly and without direct access, data x or model parameters ω that belong to other participants. An approximate inference attack on participant ℓ is defined as the adversary's ability to recover, approximately and without direct access, data x or model parameters ω that belong to other participants. The approximation quality is measured by a parameter ϵ; that is, for the inferred $\hat{x}_{\mathcal{G}}$ and $\widehat{\omega}_{\mathcal{G}}$, $\left\|\hat{x}_{\mathcal{G}} - x_{\mathcal{G}}\right\|_{\infty} \leq \epsilon$ or $\left\|\widehat{\omega}_{\mathcal{G}} - \omega_{\mathcal{G}}\right\|_{\infty} \leq \epsilon$.

7. Research methods and innovation

Federated learning involves unreliable participants. Blockchain adds incentives and transparency. The blockchain-based methods for determining evaluations, contribution weights, and incentives are reviewed herein to improve the levels of fairness and participation [16][17][18].

7.1 Blockchain-based incentive algorithm for federated learning

During the process of federated learning, malicious participants may intentionally submit incorrect gradient updates to compromise the integrity of the model training process, while free-riding or inactive nodes may contribute little to no useful data, resulting in biased or degraded model performance. Ensuring that all parties remain continuously engaged in the training process and collaborate in a fair and trustworthy manner has therefore become a critical challenge for industrial deployment scenarios. Blockchain technologies—such as consensus algorithms and smart contracts—provide inherent incentive mechanisms that can motivate participants to contribute honestly and consistently. By integrating blockchain with federated learning, it becomes possible to design blockchain-based federated learning algorithms that address these long-standing issues. The existing blockchain-enabled federated

learning approaches are reviewed herein from several perspectives, including model quality evaluation, contribution weighting, and incentive allocation schemes. Based on different incentive strategies, we categorize and analyze the representative algorithms to illustrate how blockchain can enhance the fairness, transparency, and participation incentives of federated learning systems [19][20][21].

7.1.1 Model quality evaluation-based incentive algorithm

From a technical perspective, federated learning must integrate disparate data silos into a unified collaborative framework. This process requires local data preprocessing work, followed by feature engineering of the training datasets and then quality assessments of the model inputs. In federated learning, the accuracy of model quality evaluations is a critical determinant of the effectiveness of training. Evaluation criteria—such as the data volume, data validity level, information density, and data veracity—simultaneously serve as mechanisms for achieving data monitoring and quantitative assessments.

In centralized federated learning tasks, the central aggregator must collect all locally uploaded model updates to produce a new global model. During this process, security attacks—such as model poisoning, where adversaries inject malicious samples or tamper with updates—can compromise the entire training pipeline. To sustain continuous contributions from the participants, our approach publishes task rewards within smart blockchain contracts and allocates rewards to trainers based on the results of model quality evaluation algorithms. The smart contracts thereby construct a model data transaction marketplace for federated learning, allowing honest participants to profit by completing tasks. A robust reward distribution mechanism is essential for achieving efficient multiparty collaboration; our method employs an incentive scheme that allocates rewards according to the utility that each participant generates. A federated participant's utility is defined as the marginal benefit they contribute upon joining the federated training consortium.

In general, the installment reward received by participant i in a given reward-sharing round t from the total budget $B(t)$ is denoted by $\hat{u}_i(t)$ and is computed as shown in Equation (21):

$$\hat{u}_i(t) = \frac{u_i(t)}{\prod_{i=1}^{N} u_i(t)} B(t) \tag{21}$$

where $u_i(t)$ denotes the utility that participant i derives from the reward $B(t)$. We assume that each participant's marginal contribution to the coalition (under the simplifying assumption that the coalition contains only participant i) is used to

determine the portion of the reward that the participant receives. The marginal contribution is calculated as stated in Equation (22):

$$u_i(t) = v(\{i\}) \tag{22}$$

where $v(X)$ denotes the function that evaluates the utility of the coalition X, where X represents the set of participants.

In this study, we adopt the mean squared error (MSE) as the utility function $v(X)$ for evaluating coalition X, and we propose a model quality-based evaluation incentive algorithm based on this metric. The MSE is a widely used loss function in linear regression scenarios and is minimized during the training process; it can also be employed to compare the quality levels of different model evaluations. The MSE is computed as the ratio of the sum of the squared deviations between the observed values and true values to the number of observations, yielding a per-evaluation accuracy measure, as shown in Equation (23):

$$MSE = \frac{1}{m}\prod_{i=1}^{m}(f_i - y_i)^2 \tag{23}$$

where m denotes the number of samples, and the MSE ranges over $[0, +\infty)$. A smaller MSE indicates model data with higher accuracy; when the predicted values exactly match the true values, MSE=0. A smart contract is deployed on the blockchain to record the participants, training tasks, training states, and task rewards that are involved in the federated learning process and to publish model training tasks via the smart contract. After the model data are evaluated by the model quality assessment algorithm during the federated learning process, the validated MSE values are recorded on the smart contract. Upon the completion of a training round, a preconfigured event in the smart contract triggers the payment contract to execute orderly transactions. The federated learning workflow is realized within the smart contract through three primary stages—data initialization, exchanges of locally trained model updates, and a model quality evaluation with incentive allocation—thereby enabling secure task collaboration and effective incentives, as illustrated in Figure 3.

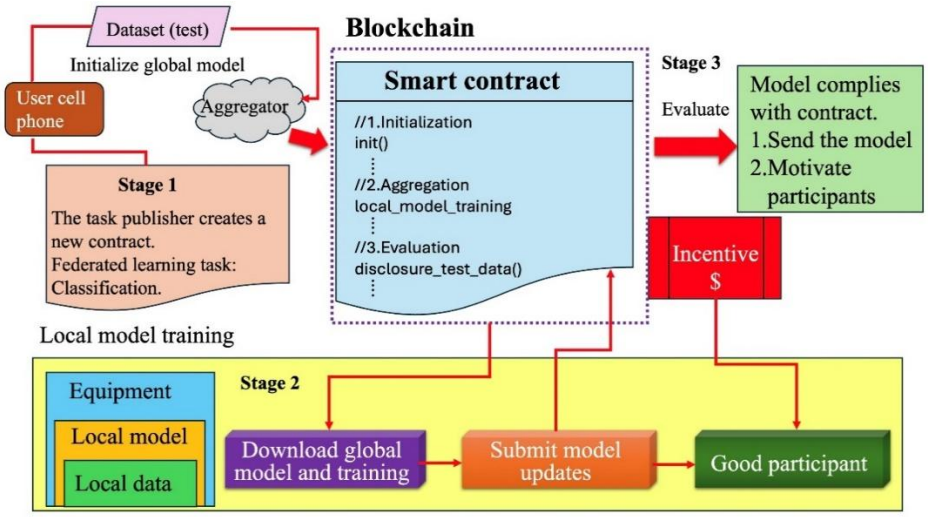


Figure 3 Model quality evaluation-based incentive process.

7.1.2 Weight-based incentive algorithm

In centralized federated learning cases, the central aggregator must coordinate the aggregation procedure across many clients and continuously broadcast updated global models to them. This design requires high network bandwidth and is vulnerable to the stability of a single server. The centralized aggregation process may also exhibit biases toward certain clients; some clients may submit maliciously corrupted model updates or leak private information through their updates, thereby compromising the integrity and user privacy of the model.

To address these issues, we propose a more secure, fair, and reliable incentive mechanism that replaces the centralized aggregator with a blockchain network. A consensus committee elected within the blockchain verifies and exchanges the local model updates received from client devices. Verified updates are recorded on the chain after the update process, and the on-chain model data are subsequently used as references for incentive weight allocation purposes. This approach reduces the computational burden of achieving consensus, mitigates model poisoning attacks, and enables more secure and efficient collaboration work.

Storing federated learning model artifacts on the blockchain ensures consistency in terms of multiparty data access and creates a transparent, auditable ledger of model data. Each block records a verifiable evidence item corresponding to a valid contribution during the training process, and global and local model artifacts are preserved on distinct block types to support traceability and accountability. As shown in Figure 4, the initial model is placed in block #0, after which the first training round commences. Devices retrieve the current global model task and perform local training; when sufficient update blocks are available, a smart contract is triggered to initiate the aggregation step. Devices elect a consensus committee to verify and exchange

their submitted model updates, and the verifier nodes upload the validated model results to a new block. Upon the completion of the round, a new global model block is generated and appended to the chain. The federated learning process relies only on the latest model block and historical block data, with the historical blocks retained solely for rollback or block verification purposes.

k denotes the number of update blocks that are required to validate each training round (the total number of updates, e.g., 100). t denotes the index of the current training round, i.e., (t=0，1，...). A model block is a blockchain block that stores the global model artifact generated during the current training round. In the formula, $\#k \times (t + 1)$ denotes the model block generated in training round t, i.e., the block number $k \times (t + 1)$. An update block is a blockchain block that stores the local model updates submitted by the client devices during the federated learning process. In the formula, $\#[t \times (k + 1) + 1,\ \big((t + 1) \times (k + 1)\big) - 1]$ denotes the set of update blocks corresponding to training round t. Table 1 presents the information for the first 100 blockchain blocks, i.e., blocks #0 through #100.

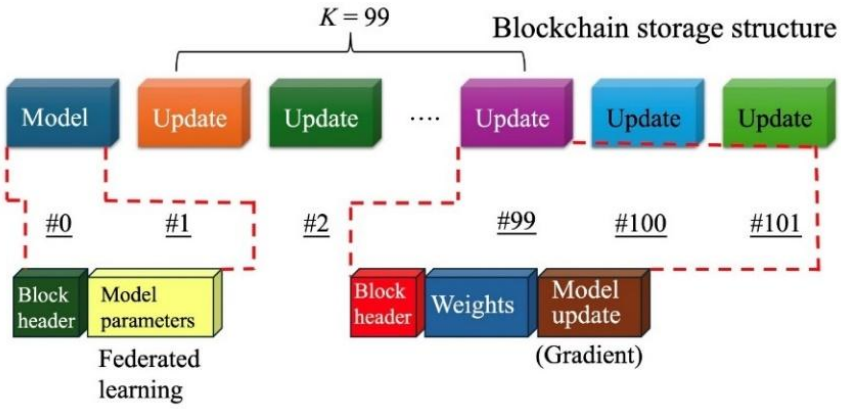


Figure 4. Block production process.

From an implementation perspective, a model block should contain the block header, the training round index t, and the global model artifact produced in that round. An update block should include the round index t, the locally updated model gradient data, contribution weight values, and other fundamental federated training metadata. The chained structure of the blockchain guarantees data immutability and determinism, and the competitive appending of correct blocks to the chain is central to achieving consensus [22][23][24].

To balance the incurred computational and communication costs while preserving agreement, an efficient and secure committee-based consensus mechanism is established. Model updates are validated prior to being written on the chain. The committee is composed of several honest consensus nodes that are responsible for

verifying device submissions and negotiating block formation processes; the remaining nodes perform local training and forward their updates to the committee. At the start of a new training round, the previous committee is not automatically reselected; instead, a new committee is elected according to the produced scoring results. The committee members apply a model quality assessment algorithm to evaluate and score the submitted model updates, and only the device updates that pass the evaluation are accepted by the committee and rewarded [25].

Table 1 Descriptions of system blocks #0–#100

Block Number	Type	Calculation Formula	Description
# 0	Model block	$\#t \times (k+1) = \#\,0$ When k = 99 and t = 0	#0 is the 0th block on the blockchain, which stores the initialization data of the federated learning model and is generated during the initialization process of federated training.
# [1, 99]	Update block	$\#\left[t \times (k+1) + 1,\ \left((t+1) \times (k+1)\right) - 1\right] = \#\,[1, 99]$ When k = 99 and t = 0	# [1, 90] is the set of blocks 1 to 99 on the blockchain (k is the round, and t is 100), i.e., the updated blocks that have passed the evaluation of the federated learning process, and are generated from device submissions.
# 100	Model block	$\#t \times (k+1) = \#\,100$ When k = 99 and t = *1*	#100 represents the 100th block on the blockchain, which is the model block generated in the first round of federated training.

7.1.3 Weight calculation

In a blockchain-based federated learning setting, the weighting coefficients for each device are evaluated according to the corresponding local training accuracy or participation frequency to account for integrity and stability. These weights are used to score and compare local updates for updating the global model. A peer denotes a device participating in the federated training procedure. Each peer trains on its local data and uploads the trained local model to the consensus nodes (Steps 1 and 2). After the committee consensus produces a new block each round (Step 4), the peers download the updated global model (Step 5) and perform the next round of local updates. This cycle is repeated until the global training process converges. A node denotes a consensus committee member that is elected from the pool of devices. The nodes are responsible for verifying the model updates uploaded by the peers and, upon reaching consensus with other nodes, generating new blocks that store the validated model data. When a node receives a model update from a device, it evaluates the update using a designated test dataset and discards any local updates that could compromise the integrity of the global model. Only model updates whose accuracies exceed a predefined threshold are accepted. After a node verifies a transaction, it signs

the validated model update and broadcasts the signed transaction to the other consensus nodes for further verification (Step 3). Once consensus is achieved, a new reliable global model is produced for the next iteration (Step 6). The entire interaction sequence is illustrated in Figure 5.

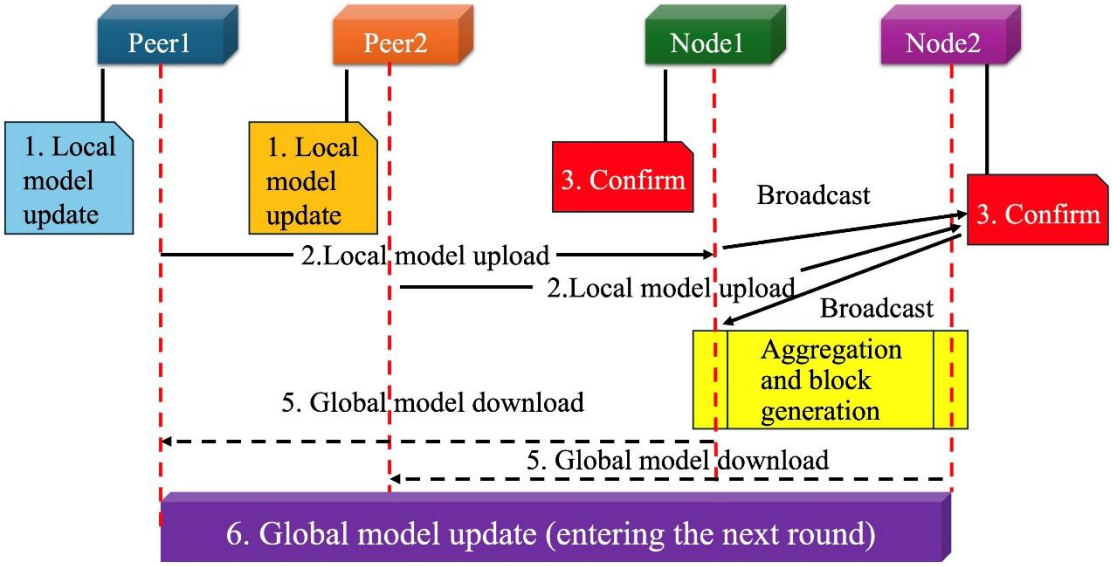


Figure 5. Model data update workflow.

The data of the MSE-based model assessment are used as the basis for the weight value evaluation task. The weight value calculation formula is as shown in Equation (24):

$$\omega^{(l)} = \omega^{(l-1)} + \prod_{i=1}^{N_D} \frac{N_i}{N_S} (\omega_i^{(l)} - \omega^{(l-1)}) \tag{24}$$

This formula represents the calculation involving local weights and global weights in the last iteration of generation l.

where ω is the weight, S is the data sample, D represents the device, N is the number of devices, and $S_k \in S$. Let $f(\omega)$ be the weight value formula based on the MSE function. In the l-th iteration, the local model data uploaded by the device node are updated as stated in Equation (25):

$$(\omega_i^{(l)},\ \{\nabla f_k(\omega^{(l)})\}_{S_k \in S_i}) \tag{25}$$

After all the local results produced in this round are validated, the global model for this

round is updated as shown in Equation (26):

$$(\omega^{(l)},\ \nabla f(\omega^{l})) \tag{26}$$

The length of the local model data submitted by each device is used as the weight value standard. After undergoing a verification by the committee, it is recorded in the update block on the blockchain. Upon the completion this round of training, the total weight value of the devices is the sum of the sizes of the model data contained in the verified update blocks. A new model block is generated, all updates related to the global model data are recorded, and the contribution of each device is quantitatively incentivized on the basis of the associated weight value. At the end of each training round, a new committee needs to be reelected to generate the next global model data block. In a distributed model training scenario, the decision-making efficiency of this committee significantly improves the performance of the entire global model training process.

7.2 Blockchain-enabled implementation for federated learning

In recent years, emerging foundational technologies—such as artificial intelligence, big data, the Internet of Things, blockchain, and cloud computing—have been widely adopted across diverse business scenarios. These technologies do not operate in isolation; rather, they can be synergistically integrated to accelerate practical deployments and foster innovations. The integration of federated learning with blockchain is one such innovative approach. By leveraging the characteristics of blockchain—trustworthiness, decentralization, immutability, traceability, and auditability—the conventional federated learning framework can be substantially enhanced.

Federated learning enables participants to collaboratively construct robust machine learning models while employing technical privacy-preserving mechanisms to protect the confidentiality of data. However, during the federated training process, the benefits of jointly processing large-scale data are not always apparent to individual contributors: data owners relinquish control over their datasets, and once data are no longer under their direct custody, their utilitarian value is substantially diminished. Although aggregating data for centralized training purposes can provide improved model performance, the allocation of such performance gains among participants remains uncertain. Therefore, federated learning requires a decentralized, distributed framework that can technically guarantee users' privacy and, under the constraints of secure data exchanges and high training efficiency, enable effective machine learning.

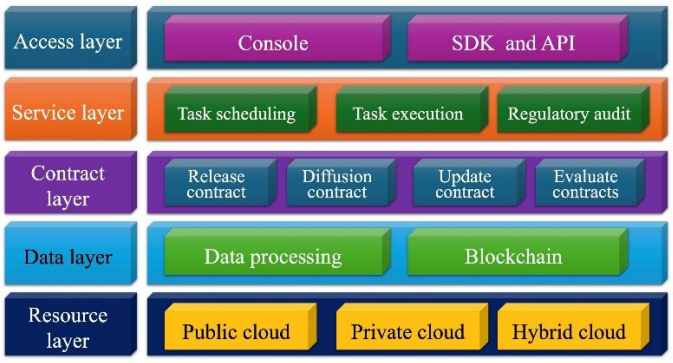


Figure 6. Component architecture.

The core essence of a blockchain-based federated learning system remains federated learning; it simply uses blockchain as an intermediary that connects the various participants. Furthermore, the involvement of blockchain gives rise to functionalities that are not found in conventional federated learning systems, such as inherent support for traceability and regulatory auditing. The architecture of a blockchain-based federated learning system can be divided into a resource layer, a data layer, a contract layer, a service layer, and an access layer, as shown in Figure 6.

The learning workflow of a federated learning system is altered by the integration of blockchain. The conventional federated learning architectures typically rely on a centralized coordination component, which is vulnerable to single-point failures and intermittent faults that can render the system unavailable or distort the output model predictions. Moreover, the lack of clear ownership of centralized services may engender mutual distrust among the participants and reduce their willingness to engage.

In a blockchain-based federated learning platform, the centralized coordination service is replaced by a decentralized architecture and smart contracts, and the participants become equal, independent parties within the decentralized network. Data exchanges and protocol enforcement operations are executed via smart contracts, eliminating a central single point of control and enabling accountability for malicious behaviors through traceability. The process flow of a blockchain-based federated learning system is depicted in Figure 7.

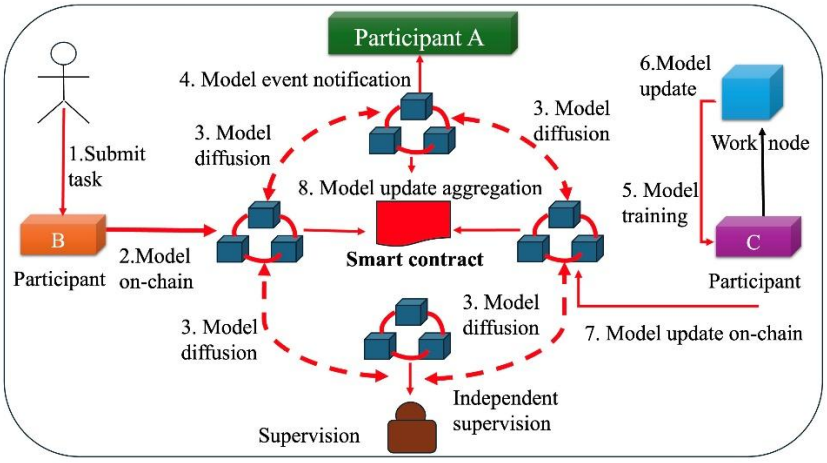


Figure 7. Flowchart of a federated learning system based on blockchain technology.

8. Experimental verification

In the experiments, our methods not only empirically validate the convergence and security of CAFSGD-RVPD, CAFSVRG-RVPD, and CAFSAGA-RVPD but also demonstrate that these algorithms are more efficient than their corresponding synchronous counterparts (i.e., CFSGD-RVPD, CFSVRG-RVPD, and CFSAGA-RVPD, respectively).

8.1 Experimental design

Our method is tested on 16 individual execution nodes on Amazon Cloud EC2. Each node has 8 Intel Xeon vCPUs, 32 GB of memory, and 25 GB of bandwidth. We use MPI for communication purposes and Armadillo v9.700.3 for efficiently executing matrix operations. In all the experiments, the normal term coefficients are $\lambda = 1e^{-4}$, and an optimal learning rate γ is selected from $\{5e^{-1}, 1e^{-1}, 5e^{-2}, 1e^{-2}, \cdots\}$. Our method assumes one lagging node whose computational speed is 1/3 slower than that of the normal worker nodes. In practice, it is common for different participants in a federated learning system to have different computational and communication capabilities.

(1) Data collection

To fully demonstrate the scalability of the asynchronous longitudinal federated learning algorithm, our method is tested on eight datasets, as shown in Table 2.

Table 2. Datasets used in the experiment

	Classification task data						Regression task data	
	Financial data		Large-scale data		Multicategory data			
	Credit1	Credit2	rcv1.binary	url	news20	rcv1.multiclass	E2006-tfidf	Year
# Classifications	2	2	2	2	20	53	----	----
# Training samples	24000	96257	677399	1916904	15935	518571	16087	463715
# Test samples	6000	24012	20242	479226	3993	15564	3308	51630
# Features	90	92	47236	3231961	62061	47236	150360	90

Two relatively small real-world financial datasets (Credit1 (UCICreditCard) and Credit2 (GiveMeSomeCredit)) are obtained from the Kaggle website, while the other six datasets are obtained from the LIBSVM website. Our method randomly splits the news20 and url datasets into training and testing datasets at a 4:1 ratio. Since the rcv1 test dataset has a larger sample size, our method uses the rcv1 test dataset for training and the training dataset for testing.

(2) Problem analysis and explanation

Our study involves comparing the results produced by the CFSVRG-RVPD and CFD-SVRG computational methods on classification and regression tasks, where the former performs almost identically. For a binary classification task, consider the following logistic regression problem with ℓ_2 norm normalization, as shown in Equation (27):

$$\min_{\omega} f(\omega) = \frac{1}{l}\sum_{i=1}^{l}\log(1+e^{-y_i\omega^T x_i}) + \frac{\lambda}{2}\|\omega\|_2^2 \tag{27}$$

where the data $x_i \in \mathbb{R}^d$ and the corresponding labels y_i are ±1. For multiclass classification problems, consider the following ℓ_2 -norm normalized polynomial logistic regression modular problem, as shown in Equation (28):

$$\min_{\omega\in\mathbb{R}^{d\times c}} f(\omega) = \frac{1}{l}\sum_{i=1}^{l}[\log\left(e^{\omega_{\cdot j}^T x_i}\right) - \sum_{j=1}^{c} Y_{ij}\omega_{\cdot j}^T x_i] + \frac{\lambda}{2}\|\omega\|_2^2 \tag{28}$$

where c is the number of categories, $\omega_{\cdot j}$ is the j-th column of ω, and the labels $y_i \in \{1, 2, \cdots, c\}$; for $Y \in \mathbb{R}^{l\times c}$, if $j = y_i$, then $Y_{ij} = 1$; otherwise, $Y_{ij} = 0$. For the regression task, ridge regression with ℓ_2 normalization is used as stated in Equation (29):

$$\min_{\omega,b} f(\omega,\ b) = \frac{1}{l}\sum_{i=1}^{l}(\omega^T x_i + b - y_i)^2 + \frac{\lambda}{2}(\|\omega\|_2^2 + b^2) \tag{29}$$

(3) Experimental procedure and review

Our asynchronous algorithm is implemented within a decentralized framework in which each participant maintains its own local dataset and model parameters. No central aggregator is employed to collect raw data, features, or gradients, since such a central node could introduce risks of user information leakages. Instead, the proposed approach uses the coordinator node depicted in Figure 2 to collect the local products computed from each participant's local data and from interactions with other participants' models. Each worker node may independently invoke the coordinator to enable asynchronous updates. The local products are aggregated in a passive manner and on demand, meaning that a participant only asks the coordinator to pull local products from other participants when it requires an update to its local parameters.

Unlike horizontal federated learning schemes that transmit gradients, our method transmits only local products, which are inherently more resistant to inference attacks. Note that if the data dimensionality in Equation (5) is *d*, then the gradient vector has a dimensionality of *d*. In contrast, the local product has a scalar dimensionality of 1 regardless of *d*. Consequently, the gradients contain substantially more information about the underlying data than the local products do, making the gradients more susceptible to reconstruction attacks. Adversaries can exploit gradient information to reconstruct the original data by formulating and solving an optimization problem, for example, by using generative adversarial networks. In our asynchronous algorithm, each worker node performs its computations independently.

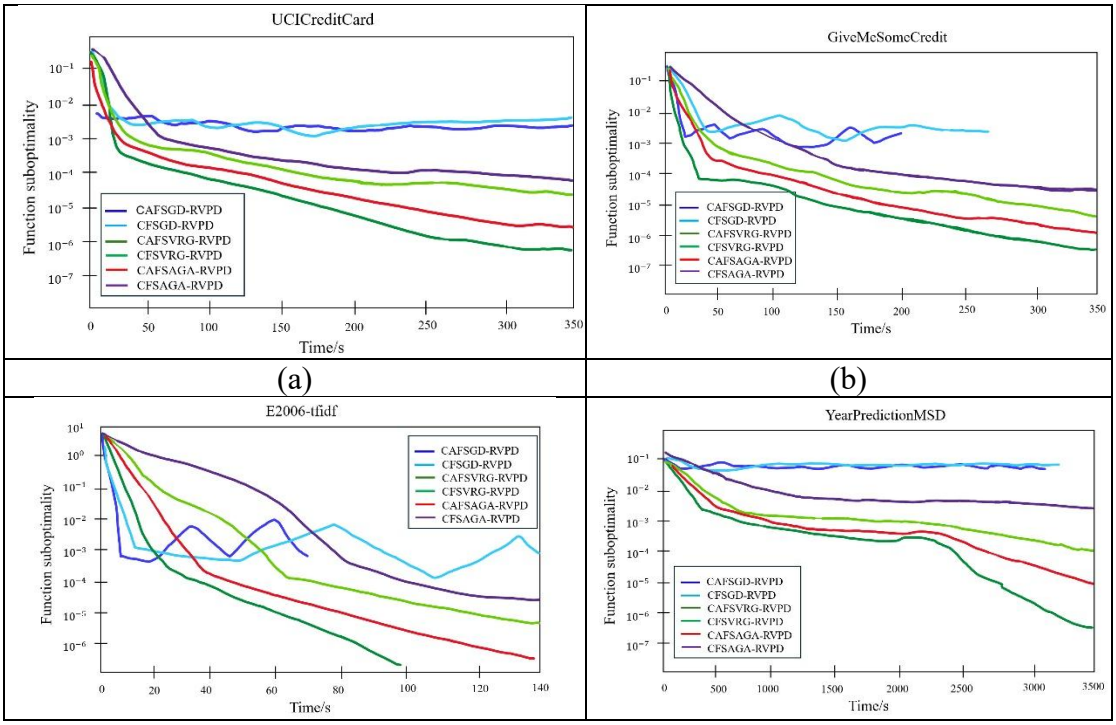
8.2 Procedures employed in this study

Our approach first involves comparing the proposed asynchronous VFL algorithm with its synchronous counterpart on financial datasets to demonstrate the applicability of the algorithm to real-world problems. In the asynchronous scheme, each participant periodically saves its local parameters at fixed time intervals for use in prediction. In the synchronous setting, when all participants operate at the same pace, each participant saves their parameters after a fixed number of iterations. The same protocol is applied in the other experiments. The original numbers of raw features in the UCICreditCard and GiveMeSomeCredit datasets are 23 and 10, respectively. Categorical features are encoded using one-hot encoding, and all remaining features are standardized. After undergoing simple preprocessing, the feature dimensionalities become 90 and 92, respectively. These experiments are conducted with four participants (i.e., four worker nodes). As shown in Figures 8a and 8b, the asynchronous VFL algorithm consistently

outperforms the synchronous algorithm.

To further illustrate the advantages of the asynchronous algorithm across different tasks, we also evaluate it on regression problems; the results are presented in Figures 8c and 8d. The regression datasets comprise E2006-tfidf (few samples, high dimensionality) and YearPredictionMSD (many samples, low dimensionality) and yield conclusions similar to those listed above. The vertical axis, i.e., function suboptimality, denotes the objective function error relative to the global optimum. The shapes of the convergence curves are determined by the chosen optimization method. In general, compared with CSVRG, CSGD achieves higher error precision, whereas the accuracy of CSAGA is comparable to that of CSVRG. The computational and communication complexity levels have the greatest effects on the resulting convergence speed. In the asynchronous setting, there is no idle waiting for slower participants, so the effective update frequency is much higher; consequently, the convergence of our asynchronous algorithm is faster when measured in wall-clock training time (rather than in training rounds). Previous experiments indicated that the proposed asynchronous VFL algorithm can more effectively address practical financial problems than can the competing method.

In this section, we further validate our method on large-scale benchmark datasets (with large sample sizes and high feature dimensionality). For the experiments conducted on rcv1.binary, news20, and rcv1.multiclass, we use eight participants, whereas the experiment implemented on the url dataset uses sixteen participants. The results, shown in Figures 9a–9d, indicate that the asynchronous variants of CSGD, CSVRG, and CSAGA outperform their synchronous counterparts across all four datasets.



(c)	(d)

Figure 8. Convergence speeds of the algorithm in classification and regression tasks.

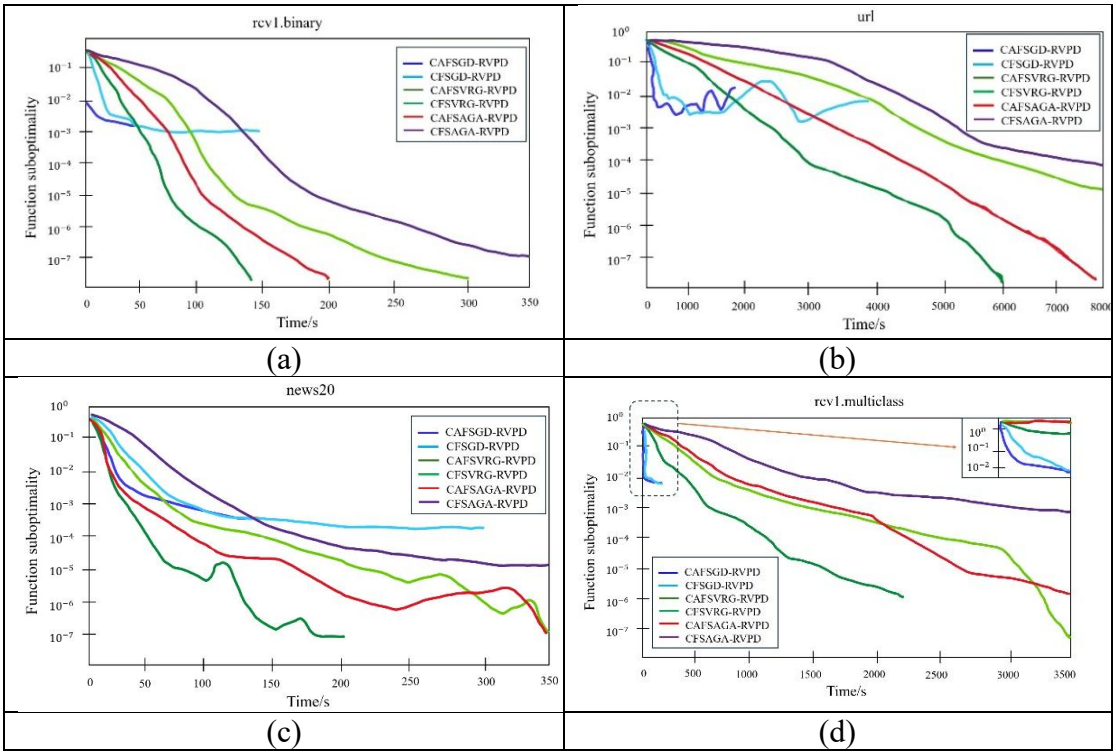


Figure 9. Convergence speeds of the algorithms in large-scale classification tasks. The upper two panels depict binary classification tasks, while the lower panels depict multiclass tasks.

Table 3 summarizes the speedups achieved by the asynchronous algorithms relative to their synchronous counterparts. The reported speedups are computed from the training time required for each algorithm to reach a predefined optimal precision level. For the algorithms based on CSVRG and CSAGA, the target precision level is set to $1 \times e^{-4}$. For the algorithms based on CSGD, the target precision level is set to $1 \times e^{-2}$ or $1 \times e^{-3}$ depending on the employed dataset.

Table 3. Speedups provided by the asynchronous algorithms

Data Collection	Accelerate		
	CSGD	CSVRG	CSAGA
UCICreditCard	2.14	2.49	1.79
GiveMeSomeCredit	2.40	2.26	2.38
rcv1.binary	2.02	2.30	2.22
url	2.11	2.64	2.05
news20	2.85	1.83	1.95
rcv1.multiclass	2.32	2.81	2.61
E2006-tfidf	2.24	2.12	2.02
YearPredictionMSD	2.26	2.15	2.38

In the synchronous algorithm, the execution time and update frequency are scaled according to the numbers of discrete time points and update steps. When a participant

lags (a straggler), its computational time is substantially longer than that of nonlagging participants, forcing the latter to incur significant synchronization delays. In contrast, in the asynchronous algorithm, nonlagging participants do not wait for stragglers to complete the current iteration; instead, they aggregate the most recent product information from the lagging parties as it becomes available. This design eliminates the synchronization idle time and yields substantial gains in the effective update frequency.

The scalability of the developed system with respect to the number of participants is illustrated in Figures 10a–10d. The lower-right panel demonstrates that the asynchronous method scales well in the presence of lagging effects. Because the synchronous algorithm requires non-lagging participants to wait inefficiently for stragglers, it cannot mitigate the straggler problem and exhibits poor scalability. The asynchronous algorithm performs strongly at small to moderate participant counts because it effectively addresses participant lag; however, as the number of participants continues to grow, the communication overhead increasingly constrains the performance gains afforded by additional participants.

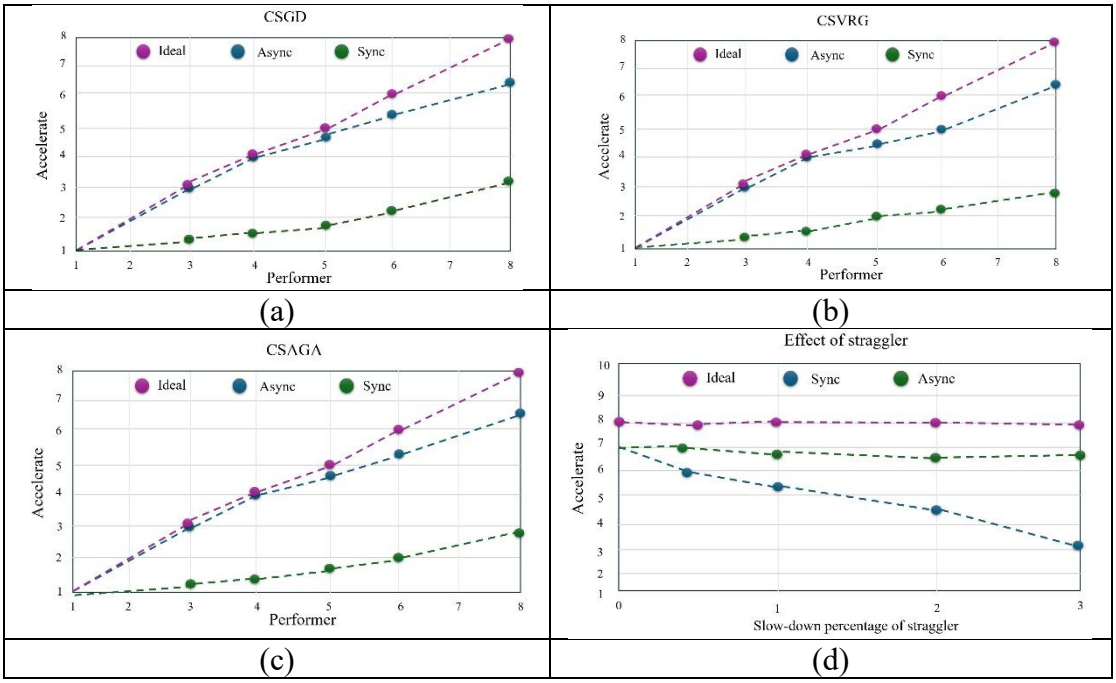


Figure 10. Results of a scalability experiment concerning the binary classification task conducted on the url dataset; the vertical axis denotes the achieved speedup.

FDML constructs a local submodel at each participant, as expressed in Equation (28), and combines the local predictions at a server node to jointly optimize the global model. Crucially, the convergence behavior of FDML depends strongly on a theoretical upper bound imposed on the communication delay; however, this bound is unknown in

practical systems. When the theoretical delay bound deviates from the actual system delay, the FDML process inevitably suffers a loss of expressive power in real deployments, as illustrated in Figure 11.

We do not compare the two methods in terms of their objective (function) values because FDML does not fully optimize the problem stated in Equation (28) but instead addresses it via local submodels. FDML only employs an algorithm based on CSGD; therefore, we compare it with our method using asynchronous CSGD. In our experiments, the performance attained with two participants is worse than that achieved with a single participant, a phenomenon also observed in Hu et al.'s logistic regression experiments. As the number of nodes increases, the performance of FDML rapidly deteriorates, whereas the performance of our algorithm remains stable. This degradation becomes more pronounced as the number of participants increases and the number of features available to each node per sample decreases, thus exacerbating the loss of representational capacity. In contrast, our algorithm scales with the number of participants because it directly optimizes the problem formulated in Equation (28).

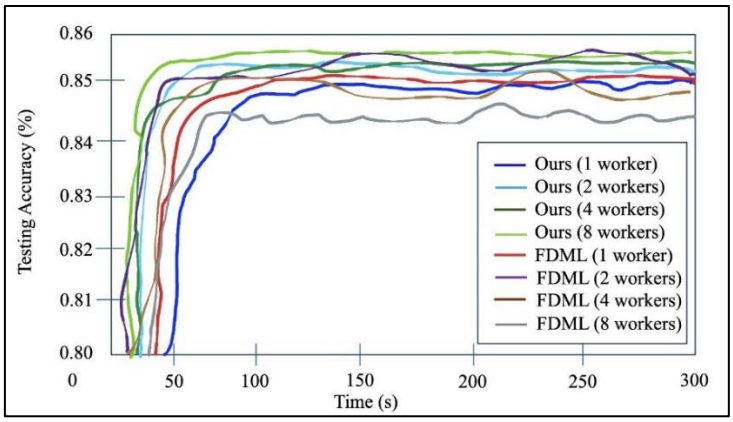


Figure 11. Comparison among different FDML computation methods in multiclass classification tasks.

9. Conclusion

In this study, an asynchronous VFL algorithm, CAFSGD-RVPD, is proposed, together with its variants, i.e., CSVRG and CSAGA. To our knowledge, these algorithms constitute the first asynchronous VFL methods that are endowed with formal theoretical guarantees. Importantly, we derive convergence rates for CAFSGD-RVPD and its CSVRG and CSAGA variants under strongly convex objective functions, and these results do not require an upper bound to be imposed on communication delays. Our analysis also establishes model-level and data-level privacy guarantees from a technical

perspective. Extensive experiments conducted on multiple vertically partitioned datasets not only validate the theoretical findings for CAFSGD-RVPD, CSVRG, and CSAGA but also demonstrate that these asynchronous algorithms are more efficient than their synchronous counterparts are.

We further implement an integrated federated learning system within intelligent embedded environments composed of heterogeneous mobile devices, and we collect and monitor the environmental parameters that are required for learning to analyze and optimize the deployment conditions; this procedure improves the practical efficiency of the federated system construction procedure. Given the widespread use of blockchain in financial markets and the robustness of its cryptographic primitives against network attacks, we leverage blockchain technology to enhance the security of the system. Specifically, we publish task rewards via smart blockchain contracts and distribute incentives to trainers on the basis of model quality evaluation metrics. The smart contract thus establishes a marketplace for completing model and data transactions during federated learning, enabling honest participants to profit by completing tasks. A fair and effective reward allocation mechanism is critical for achieving high-efficiency collaboration; our approach employs an incentive scheme that allocates rewards according to the utility contributed by each participant. In addition, we use blockchain cryptographic techniques to encapsulate federated learning artifacts, with the primary objective of preventing the theft, tampering, or destruction of sensitive federated privacy information.

Acknowledgments

This research was supported by the Department of Information Technology of Takming University of Science and Technology.

Declaration of Interests Statement

The author declares no conflicts of interest.

References

[1] S. Khan, M. Khan, M. A. Khan, L. Wang and K. Wu, “Advancing Medical Innovation Through Blockchain-Secured Federated Learning for Smart Health,” *IEEE Journal of Biomedical and Health Informatics*, vol. 29, no. 9, pp. 6482–6495, Sept. 2025.

[2] Ziyuan Yang, Wenjun Xia, Zexin Lu, Yingyu Chen, Xiaoxiao Li and Yi Zhang, “Hypernetwork-Based Physics-Driven Personalized Federated Learning for CT Imaging,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 36, no. 2, pp. 3136–3150, Feb. 2025.

[3] L. U. Khan, M. Guizani, A. Al-Fuqaha, C. S. Hong, D. Niyato, and Z. Han, "A joint communication and learning framework for hierarchical split federated learning," *IEEE Internet Things J.*, vol. 11, no. 1, pp. 268–282, Jan. 2024.

[4] E. Rabieinejad, A. Yazdinejad, A. Dehghantanha, and G. Srivastava, "Two-level privacy-preserving framework: Federated learning for attack detection in the consumer Internet of Things," *IEEE Trans. Consum. Electron.*, vol. 70, no. 1, pp. 4258–4265, Feb. 2024.

[5] W. Fan, P. Chen, X. Chun, and Y. Liu, "Madrl-based model partitioning, aggregation control, and resource allocation for cloud-edge-device collaborative split federated learning," *IEEE Trans. Mobile Comput.*, vol. 24, no. 6, pp. 5324–5341, Jun. 2025.

[6] N. Naderializadeh, J. J. Sydir, M. Simsek, and H. Nikopour, "Resource management in wireless networks via multi-agent deep reinforcement learning," *IEEE Trans. Wireless Commun.*, vol. 20, no. 6, pp. 3507–3523, Jun. 2021.

[7] H. Liu, B. Li, C. Gao, P. Xie, and C. Zhao, "Privacy-encoded federated learning against gradient-based data reconstruction attacks," *IEEE Trans. Inf. Forensics Security*, vol. 18, pp. 5860–5875, 2023.

[8] G. Zhu, Y. Deng, X. Chen, H. Zhang, Y. Fang, and T. F. Wong, "ESFL: Efficient split federated learning over resource-constrained heterogeneous wireless devices," *IEEE Internet Things J.*, vol. 11, no. 16, pp. 27153–27166, Aug. 2024.

[9] W. Yang, X. Hu, X. Zhu, R. Wu, W. Pedrycz, X. Liu and J. Huang, "FedMPS: Federated Learning in a Synergy of Multi-Level Prototype-Based Contrastive Learning and Soft Label Generation," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 37, no. 2, pp. 781–794, Feb. 2026.

[10] T. Tang, "Personalized federated graph learning on non-IID electronic health records," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 9, pp. 11843–11856, Sep. 2024.

[11] B. Wei, J. Li, Y. Liu, and W. Wang, "Non-IID federated learning with sharper risk bound," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 5, pp. 6906–6917, May 2024.

[12] B. Yan, H. Zhang, M. Xu, D. Yu, and X. Cheng, "FedRFQ: Prototype-based federated learning with reduced redundancy, minimal failure, and enhanced quality," *IEEE Trans. Comput.*, vol. 73, no. 4, pp. 1086–1098, Apr. 2024.

[13] R. Shao, P. Perera, P. C. Yuen, and V. M. Patel, "Federated generalized face presentation attack detection," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 1, pp. 103–116, Jan. 2024.

[14] D. Commey, S. G. Hounsinou and G. V. Crosby, "Post-Quantum Secure Blockchain-Based Federated Learning Framework for Healthcare Analytics," *IEEE Networking Letters*, vol. 7, no. 2, pp. 126–129, June 2025.

[15] A. P. Kalapaaking, I. Khalil, M. S. Rahman, M. Atiquzzaman, X. Yi and M. Almashor, "Blockchain-Based Federated Learning With Secure Aggregation in Trusted Execution Environment for Internet-of-Things," IEEE Transactions on Industrial Informatics, vol. 19, no. 2, pp. 1703–1714, Feb. 2023.

[16] B. Ghimire and D. B. Rawat, "Recent Advances on Federated Learning for Cybersecurity and Cybersecurity for Federated Learning for Internet of Things," *IEEE Internet of Things Journal*, vol. 9, no. 11, pp. 8229–8249, June 2022.

[17] V. Hassija, V. Chawla, V. Chamola and B. Sikdar, "Incentivization and Aggregation Schemes for Federated Learning Applications," *IEEE Transactions on Machine Learning in Communications and Networking*, vol. 1, pp. 185–196, Aug. 2023.

[18] Y. Formery, L. Mendiboure, J. Villain, V. Deniau and C. Gransart, "A Framework to Design Efficent Blockchain-Based Decentralized Federated Learning Architectures," *IEEE Open Journal of the Computer Society*, vol. 5, pp. 705–723, Oct. 2024.

[19] E. Moore, A. Imteaj, S. Rezapour and M. H. Amini, "A Survey on Secure and Private Federated Learning Using Blockchain: Theory and Application in Resource-Constrained Computing," *IEEE Internet of Things Journal*, vol. 10, no. 24, pp. 21942–21958, Dec. 2023.

[20] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 50–60, May 2020.

[21] A. Imteaj, U. Thakker, S. Wang, J. Li, and M. H. Amini, “A survey on federated transfer for resource-constrained IoT devices,” *IEEE Internet Things J.*, vol. 9, no. 1, pp. 1–24, Jan. 2022.
[22] S. Saha and T. Ahmad, “Federated transfer learning: Concept and applications,” *Intell. Artif.*, vol. 15, no. 1, pp. 35–44, 2021.
[23] Y. Kwon, J. Lee and Y. Park, “Decentralized Federated Learning on a High-Performance Computing Platform: Blockchain-Based Security and Optimization,” *Journal of Web Engineering*, vol. 24, no. 8, pp. 1231–1262, Dec. 2025.
[24] R. Myrzashova, S. H. Alsamhi, A. Hawbani, E. Curry, M. Guizani and X. Wei, “Safeguarding Patient Data-Sharing: Blockchain-Enabled Federated Learning in Medical Diagnostics,” *IEEE Transactions on Sustainable Computing*, vol. 10, no. 1, pp. 176–189, Jan.-Feb. 2025.
[25] A. Riahi, A. Mohamed and A. Erbad, “RL-Based Federated Learning Framework Over Blockchain (RL-FL-BC),” *IEEE Transactions on Network and Service Management*, vol. 20, no. 2, pp. 1587–1599, June 2023.