Mutual empowerment of distributed AI and blockchain

by

Amirreza Sokhankhosh

A thesis submitted to
The Faculty of Graduate Studies of
The University of Manitoba
in partial fulfillment of the requirements
of the degree of

Master of Science

Department of Computer Science
The University of Manitoba
Winnipeg, Manitoba, Canada
July 2025

Thesis advisor

Dr. Sara Rouhani

Author

Amirreza Sokhankhosh

Mutual empowerment of distributed AI and blockchain

Abstract

Due to their security, transparency, and decentralized trust model, blockchains have been integrated into a wide range of applications. Distributed and collaborative deep learning approaches, such as Federated Learning and Split Learning, are among the most prominent areas jointly explored with blockchain. This thesis investigates how blockchain mutually empowers distributed learning frameworks—and vice versa—through three novel contributions. First, we propose Proof-of-Collaborative-Learning (PoCL), a blockchain consensus mechanism that replaces energy-intensive mining with productive federated learning tasks, enabling secure, incentive-aligned, and resource-efficient consensus. Second, we advance SplitFed Learning (SFL) by introducing Sharded SFL for improved scalability, and Blockchain-enabled SFL (BSFL), a decentralized variant that uses smart contracts to coordinate shard aggregation and ensure model integrity. Third, we present Blockchain-enabled Personalized Federated Learning (BPFL), a framework that enhances fairness and model ownership by incorporating contribution-based personalization and a tokenized access mechanism, with theoretical convergence guarantees. Through these contributions, this thesis demonstrates how decentralized learning and blockchain can be integrated to improve security, efficiency, and fairness in collaborative AI systems.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

We are living in an era where data is not just abundant—it is foundational. As digital interactions increase, so does the volume of personal and sensitive information being generated, processed, and exchanged. As of 2020, each internet user was estimated to generate at least 1.7 megabytes of data per second [1]. Consequently, applications that rely on this data—especially those powered by Artificial Intelligence (AI)—have made data privacy one of the most urgent challenges in contemporary research [2].

The risks associated with centralized data storage—such as breaches, surveillance, and misuse—underscore the need for AI systems that prioritize not just performance, but also the privacy and security of the individuals whose data they rely on.

To address these risks, researchers have developed a variety of privacy-preserving mechanisms. These range from data anonymization techniques such as k-Anonymity [3], to cryptographic methods like Homomorphic Encryption [4], to distributed learning frameworks that eliminate the need for centralized data aggregation. This thesis

focuses on the latter.

Distributed learning refers to a class of algorithms that enable collaborative training of Machine Learning (ML) models while preserving data locally. In contrast to centralized schemes, where data is collected in a single repository, distributed learning allows participants to share only model updates. Notable approaches include Federated Learning (FL) [5] and Split Learning (SL) [6; 7], both of which significantly reduce privacy risks by transmitting intermediate computations instead of raw data. A promising hybrid is SplitFed Learning (SFL) [8], which combines the advantages of FL and SL to overcome their respective limitations.

However, despite their privacy-preserving design, most distributed learning systems still depend on a centralized server to coordinate the learning process. This reliance introduces vulnerabilities related to trust, fairness, and system robustness. In response, the research community has turned to Decentralized Learning—a novel paradigm that removes central authority by leveraging blockchain technology to manage trust, orchestrate training, and ensure transparency among participants.

This thesis explores several dimensions of decentralized AI, examining how learning protocols and blockchain infrastructures can mutually enhance one another's capabilities. Reflecting this synergy, the central theme of this work is mutual empowerment.

1.1 Research Questions

This thesis addresses the following three research questions, aimed at fostering resilient, resource-efficient, and fair collaborative learning frameworks:

1. How can blockchain consensus mechanisms be reengineered to repurpose mining computational resources for efficient and secure FL?

2. How can the integration of blockchain improve the scalability and security of SFL?

3. How can FL ensure fair model ownership while maintaining high performance?

1.2 Methodology

This thesis approaches the above questions using a unified methodology based on theoretical formulation, algorithmic design, and empirical validation.

1.2.1 Proof-of-Collaborative-Learning (PoCL)

To answer the first research question, we introduce Proof-of-Collaborative-Learning (PoCL)—a novel consensus mechanism that reimagines the mining process of blockchain as a collaborative FL task. Unlike traditional Proof-of-Work, which expends massive energy on non-productive computation, PoCL directs this computation toward training and evaluating ML models in a decentralized, multi-winner framework.

The methodology follows a three-phase pipeline: (i) model training, where miners train models on local data; (ii) peer evaluation, where miners assess each other's model predictions on unlabeled test inputs; and (iii) voting and aggregation, where smart contracts select the top-performing models based on accuracy and latency metrics. The selected models are aggregated to form a new global model, and contributors are rewarded proportionally to their performance. This framework ensures fairness,

encourages meaningful collaboration, and utilizes previously wasted computation, thereby aligning blockchain security with valuable training.

1.2.2 Advancing SplitFed Learning

The second core component of this thesis addresses the scalability and security limitations of collaborative learning by advancing the SFL paradigm through architectural and decentralized innovations. We begin by introducing Sharded SplitFed Learning (SSFL)—a novel, sharded extension of SFL. SSFL partitions the SL server into multiple independent shards, each responsible for a subset of the training workload. Clients are dynamically assigned to shards, and the resulting local models are subsequently aggregated using the Federated Averaging (FedAvg) algorithm. This layered aggregation not only mitigates server overload but also dampens imbalances caused by heterogeneous learning dynamics across shards. By redistributing server responsibilities and reducing effective learning rates through cross-shard averaging, SSFL achieves improved convergence, higher scalability, and greater training efficiency.

Building on this architecture, we propose Blockchain-enabled SplitFed Learning (BSFL), the first decentralized SFL framework. BSFL replaces the central FL server with a smart contract-governed blockchain system. A decentralized committee of validator nodes is responsible for verifying model updates, aggregating shard outputs, and distributing rewards. These processes are executed via transparent and tamper-resistant smart contracts. This design ensures model integrity, resists poisoning attacks, and eliminates central points of trust by establishing a decentralized

consensus-driven learning pipeline.

1.2.3 Personalized Federated Learning for Fairness

The third core component of this thesis tackles the fairness, transparency, and security limitations of conventional FL, particularly in enterprise or multi-stakeholder settings where model ownership and incentive alignment are critical. Although FL enables privacy-preserving model training, it allows equal access to the global model without accounting for the contributions each client made in the training process. Moreover, its reliance on a centralized server introduces vulnerabilities such as black-box implementation, single points of failure, and susceptibility to manipulation.

To address these limitations, we propose Blockchain-enabled Personalized Federated Learning (BPFL), a novel framework that integrates personalized model updates with a decentralized, token-incentivized system. At the core of our design is a novel contribution metric that quantifies each client's impact by jointly capturing two key dimensions: the improvement achieved on the client's local model and the corresponding performance gain of the global model observed on a shared synthetic validation set. This dual-focus approach ensures that both local utility and global benefit are considered in evaluating contributions.

To promote fairness, each client receives a personalized model, interpolated between its local update and the global model in proportion to its contribution. This personalization strategy helps ensure equitable model benefits and mitigates ownership disparity.

We further introduce a tokenized model access mechanism: clients with insuffi-

cient contributions may purchase access to the global model using tokens, allowing equitable participation even under resource constraints. All aggregation, personalization, and token distribution processes are managed via a permissioned blockchain, replacing the centralized FL aggregator. This decentralized orchestration eliminates the need to trust a single entity and enforces incentive alignment through verifiable, tamper-proof interactions.

Finally, we provide theoretical convergence guarantees for the proposed aggregation and personalization strategy under nonconvex, L-smooth conditions. Extensive experiments on standard benchmarks demonstrate BPFL's ability to reduce fairness disparity, maintain strong model performance, and offer a secure, incentive-compatible alternative to traditional FL.

1.3 Contribution of Authors

This thesis makes the following key contributions across the domains of decentralized ML and blockchain integration:

1. We introduce PoCL, a decentralized, multi-winner consensus protocol that replaces conventional mining with FL tasks. PoCL incentivizes useful computation by evaluating model predictions through a peer-review process and selecting high-performing models based on verifiable performance metrics. It enables secure, energy-efficient blockchain consensus while aligning incentives toward collaborative AI development.

2. We develop Sharded SplitFed Learning (SSFL), an efficient and scalable exten-

sion of the standard SFL paradigm. SSFL distributes the server role across multiple shards, improving training efficiency and robustness under client heterogeneity. Building on this, we propose Blockchain-enabled SplitFed Learning (BSFL), the first decentralized SFL framework. BSFL uses a blockchain network with smart contracts to coordinate shard aggregation, enforce update integrity, and eliminate single points of failure, enhancing transparency and resistance to poisoning attacks.

3. We propose Blockchain-enabled Personalized Federated Learning (BPFL) to address fairness and model ownership challenges in conventional FL. The framework introduces: (i) a novel metric to quantify each client's training contribution, (ii) an aggregation mechanism based on contribution scores, (iii) a personalization layer that reflects fair model ownership, (iv) a tokenized model acquisition mechanism that enables non-contributing clients to access the global model via token exchange, and (v) theoretical guarantees for convergence under standard nonconvex assumptions.

4. We conduct comprehensive empirical evaluations for all proposed frameworks by implementing both our methods and relevant baseline algorithms on widely used datasets, including MNIST, Fashion-MNIST, CIFAR-10, and CIFAR-100. To ensure consistent comparisons, we employ standard model architectures such as LeNet-5 and ResNet-18. Each framework is assessed using key performance metrics—model accuracy, convergence speed, communication overhead, and reward distribution—to validate its practicality and effectiveness. The results consistently demonstrate that our approaches offer significant improvements in

terms of security, fairness, scalability, and collaborative efficiency over existing methods.

All research work presented in this thesis was conducted by the student, Amirreza Sokhankhosh, under the supervision of Dr. Sara Rouhani. The specific contributions are as follows:

- **Amirreza Sokhankhosh** designed the research problems, developed the theoretical frameworks (PoCL, SSFL, BSFL, and BPFL), implemented all algorithms and systems, performed the experiments, analyzed the results, and wrote the thesis.
- **Dr. Sara Rouhani** provided general guidance, feedback, and supervision throughout the research process.

1.3.1 Publications

The research conducted for this thesis has resulted in the following publications:

1. Proof-of-Collaborative-Learning: A Multi-winner Federated Learning Consensus Algorithm : **Published [9]**
2. Enhancing Split Learning with Sharded and Blockchain-Enabled SplitFed Approaches : **Under Review**
3. Towards Fair Model Ownership: Blockchain-Driven Personalization in Federated Learning : **Under Review**

Chapter 2

Background & Related Works

This chapter provides the foundational background necessary to understand the core concepts and motivations behind our work. We begin by introducing the fundamental components of blockchain technology, including consensus mechanisms and smart contracts. We then present an overview of distributed learning paradigms, such as FL [5], SL [6; 7], and SFL [8], which form the basis of the systems studied in this thesis.

Following these foundational definitions, we explore the emerging field of decentralized learning—where blockchain and distributed AI intersect—and review relevant literature that motivates our contributions. Finally, we identify existing gaps in the field related to enhancing blockchain consensus with ML, improving the scalability and security of SL, and ensuring fairness in decentralized AI systems.

2.1 Blockchain Fundamentals

In this section, we briefly introduce the blockchain technology and introduce the concepts utilized throughout the thesis, including consensus mechanisms, mining procedures, and smart contracts.

2.1.1 Chain of Blocks

In 2008, Satoshi Nakamoto introduced the first blockchain network as the foundation of Bitcoin, the first cryptocurrency [10]. Following Bitcoin's rise in popularity, the blockchain architecture gained significant attention and has since been adopted across a wide range of technological domains [11; 12]. At its core, a blockchain is a decentralized ledger composed of a sequence of data blocks, where each block contains a set of transactions that can be independently verified [13; 14]. Each block in the ledger references its predecessor by storing the hash of the previous block's header, thereby ensuring the integrity and immutability of the entire chain. While a block header may contain various fields–such as the Merkle tree root, timestamp, and difficulty bits—this discussion focuses only on the components relevant to the context of this thesis. The block body contains the data stored in each block of the blockchain, which can range from validated transactions [10; 15] to ML models [16].

2.1.2 Consensus Mechanism

Consensus mechanisms in blockchains serve as fault-tolerant protocols that verify transactions and maintain agreement among all network nodes [17]. Blockchains utilize specialized consensus protocols, operated by a peer-to-peer network of nodes,

making each data block computationally challenging to produce but straightforward to verify [13]. Consensus mechanisms are intended to ensure that all honest participants in the network maintain a consistent view of the blockchain, even in the presence of faulty or malicious nodes. For example, Proof-of-Work (PoW) [10], the consensus mechanism used by Bitcoin, achieves this task by requiring miners to find a nonce that solves a difficult cryptographic hash problem through brute-force computation. This process, often referred to as mining, makes it computationally expensive to alter any part of the blockchain network. Other consensus mechanisms such as Proof-of-Stake (PoS) [15], Raft [18], and Practical-Byzantine-Fault-Tolerance (PBFT) [19] are also popular in practice. While consensus mechanisms strengthen the integrity of the blockchain ledger by making block alterations computationally or procedurally difficult, they remain vulnerable when a majority of nodes, typically more than 50%, act maliciously or collude. In such scenarios, the dominant group can enforce agreement on a falsified state of the ledger, thereby compromising the reliability and trustworthiness of the entire blockchain network.

2.1.3 Smart Contracts

Blockchain transactions can also represent computer programs, known as smart contracts, which are stored on the ledger and executed as part of transaction processing [20]. While the Bitcoin blockchain supports only rudimentary forms of smart contracts, platforms such as Ethereum [21] and Hyperledger Fabric [22] allow for the development of sophisticated, Turing-complete programs, capable of expressing any computation that can be described by a general-purpose programming language [20].

This programmability significantly expands the applicability of blockchain technology beyond cryptocurrency, enabling its integration into more complex systems and domains.

2.1.4 Blockchain categories

Blockchains can be categorized based on their access level into three different classes. For each of these categories, the architecture design as well as security precautions are different.

Public Blockchains

Blockchains such as Ethereum [21] and Bitcoin [10] are among the notable examples of this category. Public blockchains allow users to join the network without any authentication or membership inquiry. As a result, the identity of the participants is unknown, and security and privacy measurements must be taken into account at the maximum level, since any malicious node can join the network. Consequently, the consensus mechanisms of public blockchain are often more rigid and difficult to manipulate.

Permissioned Blockchains

Permissioned blockchains, such as Hyperledger Fabric and Corda, restrict participation to authenticated users who are granted access by a central authority or a predefined policy. Unlike public blockchains, these networks maintain a level of trust among participants, which allows for more efficient consensus mechanisms and faster

transaction processing.

Private Blockchains

Private blockchains are typically operated by a single organization that maintains full control over participation, data access, and transaction validation. These blockchains are not open to the public, and all nodes are internal or explicitly invited participants. The high level of centralization allows for maximum performance and minimal latency, but it also introduces trade-offs in terms of transparency and decentralization.

2.2 Distributed and collaborative Learning Paradigms

With the rapid growth of data generation and the widespread deployment of AI systems, ensuring data privacy has become a critical concern in modern AI applications. In response, distributed learning techniques have garnered significant attention, offering a way to train models collaboratively without exposing raw data. The most prominent paradigms of distributed learning include:

2.2.1 Federated Learning

FL, first introduced by McMahan et al. [5], enables collaborative model training across a network of distributed clients while preserving data locality. In a typical FL setup, a central server initializes and broadcasts a global model to participating client devices. Each client then performs local training using its private dataset and transmits the resulting model updates back to the server. The server aggregates these

updates—often through a weighted averaging scheme—to refine the global model. This cycle of model distribution, local computation, and aggregation constitutes a single communication round. By repeating this process over multiple rounds, the global model incrementally benefits from the diverse data distributions across the network, leading to improved generalization. Although originally introduced within the broader context of ML, FL has since been extensively adapted for deep learning applications in distributed environments [23; 24; 25; 26].

2.2.2 Split Learning

FL places a significant computational burden on client devices, as they are required to train the entire global model locally. To alleviate this limitation, SL was proposed [6; 7]. In SL, the global model is partitioned into two (or more) segments, with a smaller portion allocated to the clients. This design reduces the computational load on client devices by restricting their training responsibilities to only the initial layers of the model. In SL, clients train the first, lightweight model segment using their local data and transmit the intermediate activations (i.e., the output of their model's final layer) along with corresponding labels to the central server. The server then completes the forward pass using its larger model segment, computes the loss, and performs backpropagation. Finally, the server sends the gradients of the shared activation layer back to the client, enabling the client to update its model segment accordingly.

2.2.3 SplitFed Learning

Although SL offers computational advantages for clients, it incurs extended training durations due to its sequential client-server communication pattern and the need for interaction at every batch [27; 28; 29]. Additionally, SL tends to be less efficient than FL, as it lacks the benefits of aggregation strategies such as FedAvg [8].

SFL [8] was proposed to overcome the limitations of SL by integrating key elements of FL. In this hybrid framework, an additional federated server is employed to aggregate the client-side model updates using the FedAvg algorithm, resulting in notably shorter training times than traditional SL. In addition, SFL introduces the parallel training of clients, further enhancing the training time as opposed to SL.

2.3 Decentralized Learning

Although distributed learning techniques enhance data privacy by transmitting model updates rather than raw data, they remain vulnerable to critical issues related to stability, fairness, and security due to their reliance on a centralized server:

- **Fairness:** In most distributed learning setups, the central server operates as a black-box entity, potentially leading to biased aggregation strategies. In consequence, it may favor certain clients over others, undermining both the equity and effectiveness of the training process.

- **Security:** Centralization introduces notable security vulnerabilities. A malicious or compromised server, either acting independently or through collusion, can degrade global model performance by injecting harmful updates. Moreover,

despite the absence of direct data sharing, the server may launch privacy attacks such as Membership Inference Attacks (MIA) [30], revealing sensitive information about clients' training data using only black-box access. Furthermore, the absence of effective evaluation and incentivization mechanisms for client submissions enables malicious behavior, allowing clients to contribute untrained or deliberately harmful model updates. This undermines the integrity of the aggregation process and degrades the overall performance of the global model.

- **Stability:** The central server represents a single point of failure. Any malfunction, compromise, or dropout can disrupt the training process entirely, potentially nullifying the contributions made by participating clients up to that point [11].

Replacing the centralized server with a decentralized blockchain network resolves the problems mentioned above. Through the integration of blockchain in distributed AI algorithms, the responsibilities of the central server can be seamlessly handled by smart contracts. The decentralized learning framework is a term often used for families of algorithms that integrate blockchain with distributed AI technologies. Blockchain-enabled Federated Learning is the first among these [11]. The decentralized architecture of blockchain technology addresses the server-related limitations of FL by removing the need for a central coordinating entity. Moreover, blockchain's native incentive mechanisms help promote honest client participation by rewarding contributors based on the quality of their model updates. In this context, Kim et al. [16] proposed BlockFL, the first blockchain-integrated federated learning framework. Building on this foundation, Li et al. [31] developed a blockchain-based federated

learning design that employs a committee-driven consensus protocol.

The application of blockchain in FL extends beyond committee-based consensus and incorporates a range of enhancements, including the integration of differential privacy techniques [32; 33; 34], Layer 2 (L2) scaling solutions [35; 36], and Zero-Knowledge Proofs (ZKPs) [37; 38]. These methods are collectively designed to improve the privacy, scalability, and security of blockchain-supported federated learning systems. Practical applications of FL integrated with blockchain are across a range of domains, such as the industrial Internet of Things (IIoT) [39; 40; 41], intelligent healthcare systems [42; 43], and wireless communication infrastructures [44; 45; 13].

2.4 Research Gaps

In this thesis, we present novel solutions addressing key limitations in various aspects of Decentralized AI. Our contributions are organized into two primary categories: *Blockchain for AI* and *AI for Blockchain*, reflecting two complementary directions in which the integration of these technologies yields mutual benefits. Our *AI for Blockchain* approach reimagines blockchain consensus mechanisms as a means of collaboratively training ML models by leveraging the computational resources typically used for mining. Conversely, the *Blockchain for AI* methods focus on improving the scalability and security of SFL, and enhancing fairness in FL through personalized model aggregation. The subsequent subsections review relevant recent advancements in each of these research areas.

2.4.1 Enhancing Consensus Mechanism

Nakamoto's introduction of the Proof-of-Work (PoW) consensus mechanism in Bitcoin [10] established a computation-intensive process for maintaining blockchain integrity. In PoW, miners compete to identify a nonce that satisfies specific hash conditions to validate and append new blocks. While effective, the significant computational cost associated with PoW has motivated extensive research into more sustainable and resource-efficient alternatives [46].

In response, various consensus protocols have been proposed that reduce the energy demands of PoW while preserving security guarantees [17]. One prominent alternative is Proof-of-Stake (PoS), introduced by Nguyen et al. [15], which designates block validation rights based on the stake held by participants. Despite improving efficiency, PoS has raised fairness and centralization concerns, prompting refinements across several studies [47; 48; 49; 50; 51; 52].

Several hybrid or specialized mechanisms have also emerged. Proof-of-Work-Time (PoWT) [53] incorporates a temporal dimension to PoW, aiming to reduce energy waste by factoring in block time during validation [17]. Proof-of-Burn (PoB) [54] incentivizes miners to irreversibly destroy coins by transferring them to unrecoverable addresses, thereby acquiring virtual mining privileges. Similarly, Proof-of-Space (PoSpace) [55] replaces computational effort with storage commitments, requiring participants to allocate memory as proof for validation.

Building on these developments, Proof-of-History (PoH) [56] enhances blockchain throughput by cryptographically verifying the sequence of events, ensuring time consistency without requiring conventional timestamping. Proof-of-Activity (PoA) [57]

integrates PoW and PoS, initiating block creation through mining and finalizing it through stake-based selection, thereby balancing energy efficiency with security. Finally, Snow White [58] refines PoS models by introducing formal security guarantees and promoting fair validator participation even under adversarial conditions.

Several studies have explored substituting energy-intensive Proof-of-Work (PoW) puzzles with tasks that serve productive purposes [59]. For instance, Proof-of-eXercise (PoX) [60] and Proof-of-Useful-Work [61] redirect mining resources toward scientific computing and the solution of complex polynomial problems, respectively. Primecoin [62], on the other hand, utilizes computational efforts to discover sequences of prime numbers. Our focus lies on consensus mechanisms that embed ML processes into their foundational design.

Bravo-Marquez et al. [63] introduced Proof-of-Learning (PoL), where participants known as "trainers" submit ML models that are assessed by a group of "validators." Rewards are granted to the trainer whose model performs best. In a related approach, Proof-of-Federated-Learning (PoFL) [59] involves miners collaboratively training models within distinct mining pools to compete for rewards. However, this structure grants an early and persistent advantage to the winning pool, as their superior model is not shared across pools. Furthermore, PoFL does not yield a unified global FL model, since each pool independently trains its own architecture. A comparable limitation is found in Proof-of-Training-Quality [64], which applies FL within isolated local committees.

In summary, current consensus mechanisms lack the capacity to support a fair, secure, and globally consistent FL process. Chapter 3 introduces a novel framework

designed to address this shortcoming by proposing a decentralized mechanism that ensures equitable participation and validation in FL.

2.4.2 Decentralized Split Learning

In Chapter 4, we explore methods to enhance the scalability, security, and efficiency of SFL. The subsections below provide an overview of prior research addressing these issues.

Scalibility

The split architecture inherent to SL and SFL presents a fundamental scalability limitation [65]. In parallel SL, the server-side model is updated more frequently than the client-side segments, leading to a disparity in learning progress between the two. This imbalance can degrade training efficiency, particularly when the server and client updates are processed independently. To mitigate these issues, LocFedMix-SL was introduced, which employs local regularization on client models and augments the clients' smashed data to enhance the server's learning process [65].

Subsequent research has proposed further improvements. For example, Pal et al. [66] suggested using distinct learning rates for the server and clients and implemented gradient broadcasting to harmonize updates across clients. More recently, Mix2SFL integrated these techniques, resulting in improved model accuracy and communication efficiency [67].

Despite their contributions to scalability, these methods do not adequately address the computational and communication burdens placed on the central server, which

can hinder overall system performance. To overcome these challenges, we introduce SSFL, a novel sharding-based framework that improves the scalability, stability, and training efficiency of SFL.

Security

The integration of blockchain technology with SL remains a relatively underexplored domain. BlockFeST, introduced by Batool et al. [68], merges blockchain with both FL and SL paradigms, shifting most of the computational load to an SL server in order to reduce the burden on FL clients. In a related effort, Sai [69] proposed a blockchain-based SL framework that utilizes smart contracts to dynamically assign servers and clients for each training round. Although these methods enhance fairness and improve security through decentralization, they still suffer from centralization bottlenecks and limited scalability.

To the best of our knowledge, this thesis is the first to introduce Blockchain-enabled SplitFed Learning, a novel framework that systematically addresses the security vulnerabilities, fairness constraints, and performance inefficiencies inherent in conventional SL approaches.

2.4.3 Fairness in Decentralized AI

Over the years, considerable efforts have been made to introduce various notions of fairness in FL. The aspect of fairness we are interested in is named Collaborative Fairness, first introduced in Collaborative Fair Federated Learning (CFFL) [70], referring to fair global model access in proportion to each client's reputation.

CFFL tackles this problem by deriving each participant's reputation from the accuracy of a subset of its uploaded gradients on a public validation set. This reputation reflects how informative a client's gradients are and is utilized to achieve fairness by limiting access: low-reputation participants get fewer gradient updates. Nevertheless, all clients receive a singular global model; hence, the unjust model ownership issue remains unsolved.

A similar reputation scheme has been used in Robust and Fair Federated Learning (RFFL) [71], where each client's reputation is iteratively updated by measuring the cosine similarity between its uploaded gradient and the current aggregated update, and those scores are then used both to weight the gradient aggregation and to sparsify the portions of the global update that each client downloads. While this dual-use of reputation in RFFL does improve robustness against poisoned or low-quality updates and aligns the "reward" of gradient access with measured contribution, it still delivers a single global model to everyone. As a result, true model ownership remains undifferentiated, and incentives for fair participation are not fully enforced at the level of each client's received model.

In chapter 5, we employ the concept of contributions in measuring and enforcing fairness. A similar notion for contribution-aware aggregation has been proposed in FedCon [72]. In this algorithm, weights are initialized using a KL-divergence–based "data quality" term plus data volume, then refines them each round via (i) Monte Carlo–approximated Shapley values on model performance and (ii) a cosine-similarity adjustment, with a linear decay schedule blending initial and Shapley-based weights. This pipeline is explicitly designed to overcome non-IID splits and to smooth out

drastic weight jumps. Nevertheless, a single global model is propagated between all clients, which diminishes fair model ownership.

Chapter 3

Repurposing Computation Power

3.1 Problem Statement

The decentralized nature of blockchain technology, the incentive mechanisms supporting blockchain networks, and smart contracts—which enable programmable and automated transactions—have all contributed to blockchain's explosive expansion. Blockchains, especially cryptocurrency networks, use consensus mechanisms such as Proof-of-Work (PoW) to prevent double spending of tokens; nevertheless, this mechanism requires substantial computational power to preserve data integrity and security. The most well-known use of PoW, Bitcoin mining, has an energy demand currently equal to the yearly energy usage of nations like Poland [73]. These serious environmental issues emphasize the need for more effective consensus mechanisms to retain blockchain's advantages while mitigating its environmental impact.

PoW requires miners to solve computationally expensive hash puzzles by iteratively searching for a valid nonce. While effective in securing the blockchain, this

process suffers from two core inefficiencies:

- The computation involved in solving PoW puzzles is inherently unproductive; it serves no purpose beyond enforcing difficulty in block manipulation.
- Only the successful miner's computation is utilized and rewarded, while the efforts of all other miners are discarded, wasting massive computational resources without any residual benefit.

These drawbacks have motivated researchers to explore more meaningful alternatives to PoW. Some proposals seek to embed useful computational tasks within the consensus process, such as deep learning training or scientific computing—in what has been termed proof-of-useful-work [59; 60; 61; 62]. Other studies [15; 53] have also suggested alternatives with less computational requirements, using various consensus models that focus on efficiency and lower energy consumption.

In parallel, FL, introduced by McMahan et al. [5], has emerged as a promising paradigm for privacy-preserving machine learning. FL enables collaborative model training across decentralized clients without transferring raw data to a central server. Despite these benefits, privacy concerns persist due to possible leakage from model updates and the presence of a central aggregator. Furthermore, the lack of robust incentive mechanisms may discourage honest participation.

To address these concerns, researchers have proposed blockchain-integrated FL frameworks. Notably, BlockFL by Kim et al. [16] introduces smart contracts for verifying and incentivizing model updates, thereby decentralizing trust. Since then, numerous extensions have applied this fusion to domains such as IoT, medical imaging, and cognitive networks [39; 40; 41; 42; 43; 44; 45; 74].

Among these, Proof-of-Federated-Learning (PoFL) [59] integrates the FL architecture in the consensus mechanism of blockchains. In PoFL, requesters send Deep Learning (DL) tasks along with a list of possible data providers. These tasks are distributed among distinct pools of miners, with each pool selecting a DL model for the task. In each pool, a miner is selected as the leader and serves as the central server in the traditional FL paradigm, while the others operate as clients. However, there are a few major drawbacks to this approach:

1. **Limited collaborative scope:** In PoFL, FL is executed within isolated mining pools, typically composed of a small subset of miners. This restriction reduces the diversity and representativeness of the data available for training, thereby limiting the generalizability and performance of the resulting models. In addition, besides the winning pool, the training contributions of all mining pools are lost since no aggregation step is performed between mining pools.

2. **Lack of model-sharing fairness:** The framework encourages competition among pools to produce the best-performing model; however, the winning global model is not shared with other pools. This disparity significantly affects miners who join the competition in the middle or final rounds.

3. **Requester-specific constraints:** In most practical scenarios, requesters specify both the model architecture and associated datasets. PoFL's design, which assumes flexibility in model and data distribution across mining pools, often fails to accommodate such tightly defined training requirements, limiting its applicability in real-world deployments.

To address the limitations of existing FL-based consensus mechanisms, this chapter introduces Proof-of-Collaborative-Learning (PoCL)—a novel, fully decentralized, multi-winner consensus protocol for FL. PoCL improves model evaluation and fairness by leveraging a distributed network of miners for collaborative assessment. In each training round, miners exchange unlabeled test samples to evaluate the predictive performance of peer models. Participating miners generate predictions on these samples, which are then assessed based on two key criteria: accuracy (measured via loss functions) and timeliness (measured by prediction latency).

Using smart contracts, the system aggregates evaluations and selects the top K performing miners based on a voting mechanism. These selected miners are designated as round winners and contribute their models to form an updated global model. Their contributions are rewarded proportionally through transparent, automated incentives governed by smart contracts. PoCL thus mitigates issues of unfair competition, lack of model sharing, and single-winner inefficiencies prevalent in prior frameworks—achieving better incentive alignment, fairness, and collaborative model improvement.

3.2 Design

In this chapter, we extend the principles of FL to develop a novel consensus mechanism integrated into a decentralized FL framework, called Proof-of-Collaborative-Learning (PoCL). Traditional FL systems involve a central server coordinating multiple clients, each training a shared global model on private local data and returning model updates for aggregation. The server aggregates these updates into an improved

global model at the end of each training **round**.

To decentralize this process and enable consensus through collaboration, we introduce a set of steps that must be followed in each round of PoCL. Before delving into the architectural workflow, we describe the key components involved in our system.

3.2.1 Components

To effectively implement our FL-based consensus mechanism, we define the following components within our network:

- **Administrator**: This component is responsible for altering user-defined values, including the number of rounds, number of winning miners, and deadlines for each step, or any other parameter or hyperparameter in the system. The administrator is also responsible for notifying the miners about the details of each step. To ensure that this role does not compromise the network's decentralization, the Administrator's functions can be implemented through decentralized governance mechanisms. For instance, changes proposed by the Administrator could require approval through a distributed consensus process among selected stakeholders or be automated through smart contracts that execute based on predefined rules agreed upon by the network participants.

- **Requesters**: In the proposed network, individuals who submit deep learning tasks are referred to as requesters. Each task submission, or request, includes a deep learning model intended for global training, along with a curated list of publicly accessible datasets designated for that purpose. To address potential privacy concerns associated with distributing private data among multiple

miners, requesters must select strictly from publicly available datasets for the training process. Submitted requests are stored in a queue, from which one is selected in each consensus round to determine the model to be globally trained.

- **Miners**: Miners are responsible for contributing new blocks to the blockchain by participating in the training of global deep learning models, generating predictions on test data provided by other miners, and evaluating peer predictions through a voting process. To become winners in each round, miners compete to produce the most effective model within a time frame determined by the administrator. The rewards distributed to winning miners are proportional to the value of their contributions, measured by the average divergence between their locally trained model weights and the previous global model.

- **Aggregator**: An off-chain entity responsible for coordinating the aggregation of models submitted by the winning miners. Depending on the preferences specified by the requester, the aggregator is capable of switching between various FL strategies. In addition to model aggregation, it evaluates each winner's contribution and communicates the results to the blockchain to facilitate appropriate reward distribution.

- **Users**: Like other participants in a blockchain network, users submit transactions intended for inclusion in the blockchain. These transactions are temporarily held in a transaction pool, from which miners choose a subset to include in the next block. To incentivize faster processing, users may attach additional fees, thereby increasing the likelihood that miners prioritize their transactions.

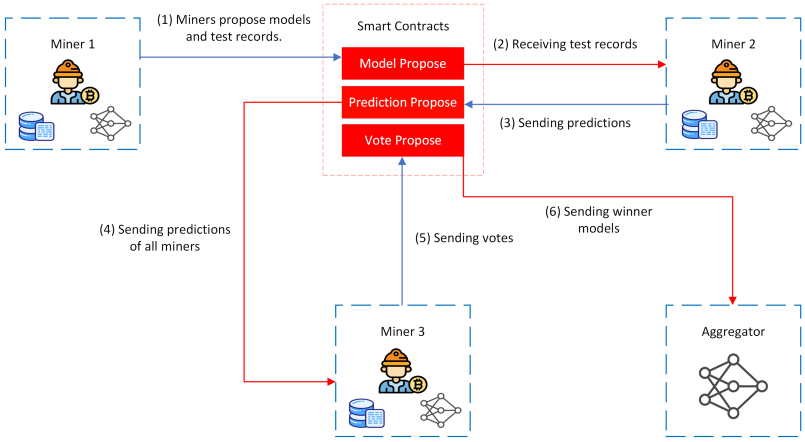


Figure 3.1: Proof-of-Collaborative-Learning (PoCL) Design.

Furthermore, we assume that all peers in the blockchain network are equipped to store and execute smart contracts. Under these assumptions, we define a structured sequence of actions to be performed in each round to reach consensus through globally validated FL. The overall workflow of the proposed framework is illustrated in Figure 3.1 and detailed in Algorithm 1.

3.2.2 Model Proposal

Each consensus round begins with the selection of a global deep learning model from the request queue, which is then broadcast to all miners. Upon receiving the model, miners select a verified subset of transactions and initiate the local training process. After training the model on their private data, each miner constructs a model proposal block that includes a hash of the locally trained model and a curated set of unlabeled test samples. These test cases, distributed across clients, are carefully chosen to reflect the overall data distribution while preserving privacy. This evaluation step ensures a consistent and fair comparison of local models under similar conditions.

At the start of each round, miners are informed of a designated time window known as the model proposal deadline. Within this period, miners must select transactions, train the assigned global model, and submit their model proposal block. Failure to meet this deadline results in exclusion from the remainder of the consensus round. The duration of this deadline is determined by the administrator and can be tailored to the specific requirements of different systems, miners, or model complexities. In scenarios where the global model demands extended training to achieve satisfactory performance, this parameter can be adjusted accordingly, offering flexibility to system operators.

3.2.3 Prediction Proposal

To identify and aggregate the highest-performing models, an evaluation process is required. In our FL framework, we implement an innovative and decentralized evaluation protocol that allows for the assessment of local models trained on miners' devices. This approach ensures that only the most effective miners contribute to the global model while also safeguarding the privacy of their local models.

After passing the model proposal deadline, the submitted test records are collected and sent to miners who participated in the model proposal step. The miners predict the given test records by feeding the inputs to the model. Afterward, they forward the predictions to a smart contract, managing all miner predictions.

Once the model proposal deadline has passed, the submitted test records are gathered and distributed to the miners who participated in the model proposal phase. These miners generate predictions by entering the test records into the trained model.

The predictions are then forwarded to a smart contract that manages all miner predictions.

To prevent miners from inferring the correct labels of the test records using methods other than their trained models, we impose a deadline on the prediction proposal phase. This deadline is carefully set to allow only enough time for a forward pass of the global model. Any proposals submitted after this deadline will be rejected to

Algorithm 1 The general workflow of the framework

```
1:  class Miner:
2:      function mine():
3:          miningTrx ← blc.getTrx(self.id)
4:          testRecords ← getTestRecords()
5:          model.fit(localData)
6:          modelHash ← md5(model)
7:          blc.submitModel(modelHash, testRecords)
8:      function predict():
9:          testRecords ← blc.getTestRecords()
10:         preds ← model(testRecords)
11:         blc.submitPreds(preds)
12:     function vote():
13:         preds, times ← blc.getPreds(self.id)
14:         losses ← lossFn(preds, testLabels)
15:         votes ← sort(losses, times)
16:         blc.submitVotes(votes)
```

maintain the integrity of the evaluation process.

The administrator is also responsible for setting the prediction proposal deadline, which may differ across systems depending on specific scenarios. Typically, the procedure involves running an experiment to measure the average time miners take to perform a forward pass of the global model and submit their results. This empirically obtained time can then be used as the foundation for determining the prediction proposal deadline.

3.2.4 Vote Proposal

The vote proposal phase plays a critical role in distinguishing between honest and adversarial miners. In this phase, miners evaluate the predictions made by their peers in the previous step based on two key factors: the loss value associated with the predictions and the time taken to submit them. First, they rank the predictions from most to least accurate. In cases where predictions are equally accurate, preference is given to those submitted earlier. This approach incentivizes miners to adhere to the guidelines for the prediction proposal phase and submit their results promptly. Additionally, it strengthens the system's defense against potential attacks, as outlined in the discussion section.

Algorithm 1 The general workflow of the framework (Continued)

```
class Blockchain:
    function start():
        for each miner in miners do:
            notify(miner, "mine")
        end for
        setTimeout(getPreds, modelDeadline)
    function getPreds():
        for each miner in miners do:
            notify(miner, "predict")
        end for
        setTimeout(getVotes, predDeadline)
    function getVotes():
        for each miner in miners do:
            notify(miner, "vote")
        end for
        setTimeout(selectWinners, voteDeadline)
    function selectWinners():
        votes ← gatherAllVotes()
        winners ← votes[:k]
        runTrx(winners)
        rewards ← agg.combine(winners)
        giveRewards(winners, rewards)
```

3.2.5 Winner Selection

A Chaincode aggregates all submitted votes and identifies the top K miners with the highest vote counts as the winners of the round. Their trained models are then sent to the aggregator, which applies the FL combination algorithm, such as Federated Averaging (FedAvg) [5], to combine the models.

The aggregator ensures the integrity of each winning model by checking its hash against the one proposed in the initial step, verifying that the model has not been altered since its proposal.

3.2.6 Block Creation

In the *Model Proposal* step, each miner is allocated a set of validated transactions to process. A smart contract then collects the transactions from the winning miners, combines them into a single block, and appends it to the blockchain.

3.2.7 Reward Mechanism

In this framework, we propose a fair reward distribution mechanism that compensates winning miners based on the importance of their contributions to the global model in the previous round. The global model is denoted as $M_G = [\tilde{W^1}, \tilde{W^2}, ..., \tilde{W^L}]$ and the local model of the i-th miner is represented as $M_i = [W^1, W^2, ..., W^L]$. In these expressions, W^1 and $\tilde{W^1}$ represent the weights of the first layers of the local and global models, respectively, while L denotes the total number of layers in the model. Since the sizes of the layers may differ, the reward for the winning miner i is computed using a formula that considers these variations:

$$r_i = \frac{1}{L} \sum_{l} \frac{1}{N_l} \sum_{n} |W_n^l - \tilde{W}_n^l|, \tag{3.1}$$

$$\alpha_i = \frac{\exp(r_i)}{\sum_{j=1}^{K} \exp(r_j)}, \tag{3.2}$$

$$R_i = \alpha_i . R_{\text{total}}. \tag{3.3}$$

In simpler terms, we calculate r_i as the average contribution of each winning miner to the current global model and distribute the total round reward R_{total} among the winning miners based on their softmaxed contribution scores. At the end of each round, the aggregator sends this information to the smart contract, which then issues rewards to the winning miners. In the results section, we demonstrate how this reward mechanism improves the fairness of our proposed framework.

3.2.8 Repetition

The consensus process described above is repeated through multiple rounds until the global model converges. Convergence can be monitored using a decentralized validation metric, such as the average improvement in model performance on a shared validation set over recent rounds. Both the committee and the participating miners can implement early stopping criteria, halting the training phase if no significant performance improvement is seen over a set number of rounds. This ensures computational efficiency while maintaining the quality of the model.

3.3 Discussions

3.3.1 Security Concerns

A potential threat to our FL-validated consensus mechanism lies in the possibility of miners bypassing training requirements. Specifically, some miners may attempt to deceive others by generating predictions using alternative supervised or unsupervised algorithms, such as k-Nearest Neighbors (KNN), rather than properly training the global federated model. In this section, we analyze two threat scenarios involving such behavior and argue that in both cases, adversarial miners will not gain a competitive advantage. Furthermore, our simulation results in Section 5.6 empirically validate this claim by demonstrating the limited effectiveness of KNN-based attacks across varying data sizes.

We begin by defining the training dataset as $\mathcal{T}_{train} = \{x_i\}_{i=1}^{N_{train}}$ where $x_i \in \mathbb{R}^D$.

High-dimensional Setting

When the feature dimension D is large, especially for modalities such as image or audio data, non-parametric methods like KNN are generally outperformed by deep learning models. This performance gap arises due to the curse of dimensionality, where distance metrics become less meaningful as dimensionality increases [75]. Consequently, adversaries attempting to shortcut training by using KNN will generate lower-quality predictions, reducing their chances of securing votes and rewards in the PoCL mechanism.

Low-dimensional Setting

In contrast, when the feature dimension D is small, the performance of KNN may closely match that of neural networks. In such cases, we intentionally select a simple, fully connected architecture for the global model, since advanced architectures like convolutional neural networks (CNNs) offer diminishing returns on low-dimensional data. This ensures that model complexity remains appropriate for the data while keeping inference efficient.

Let N_{test} and N_{train} denote the number of test and training examples, respectively. The time complexity of generating predictions using KNN is:

$$T(N_{train}, N_{test}, D) = O(N_{test} \cdot N_{train} \cdot D + N_{test} \cdot \log N_{test}), \tag{3.4}$$

Where the first term accounts for distance computations and the second for sorting nearest neighbors.

By contrast, the time complexity of a forward pass through a fully connected neural network is:

$$T(N_{test}, D, H_{\max}) = O(N_{test} \cdot D \cdot H_{\max}), \tag{3.5}$$

Where $H_{\max}$ denotes the size of the largest hidden layer. This forward pass is highly parallelizable and can be efficiently accelerated using GPUs. Unless $H_{\max}$ is extremely large, the neural network's inference time is generally faster than that of KNN, particularly when N_{train} is large.

Security implication: In summary, adversarial miners are disincentivized from using simpler models like KNN in both high- and low-dimensional regimes. In the

former, they suffer from inferior prediction accuracy; in the latter, they may match model performance but incur greater computational cost and lower scalability. Since the PoCL voting mechanism favors accurate and faster predictions, honest participation in global model training offers the highest expected reward, aligning system incentives with collaborative learning goals.

3.3.2 PoCL vs PoW

PoCL repurposes the computational power in blockchain systems by redirecting it from solving arbitrary puzzles, as in PoW, to contributing to collaborative machine learning tasks. While PoCL retains some structural similarities with PoW, it also introduces significant departures that warrant discussion.

Similarities

- **Block Creation Through Mining:** Like PoW, PoCL adopts a mining-based framework for block creation and preserving the integrity in the blockchain ecosystem. This sets it apart from consensus mechanisms such as Raft [18], PBFT [19], and most Proof-of-Stake variants [15; 47; 48; 49; 50; 51; 52], which do not rely on a mining paradigm.

- **Transaction Flow Architecture:** Both PoCL and PoW follow the traditional Order-Execute model of transaction processing, where transactions are first ordered (via a consensus mechanism) and then executed sequentially by all peers [22]. This pattern remains dominant across most blockchain platforms. However, alternative transaction flows exist; for instance, Hyperledger

Fabric employs an Execute-Order-Validate model, which fundamentally alters the block generation process.

Differences

- **Nature of Competition** The core difference between PoCL and PoW lies in the objective of their mining competitions. In PoW, miners race to solve a cryptographic puzzle by discovering a valid nonce for a SHA-256 hash, making the competition inherently time-based—the first to solve the puzzle is declared the winner. In contrast, PoCL reframes this competition as a collaborative machine learning task, where success is measured not by speed alone, but also by the quality of the trained model. Since rapid training does not guarantee high model performance, PoCL must evaluate both model accuracy and training efficiency. Moreover, to sustain consistent transaction flow, the duration of each training round must be carefully calibrated–miners are given a fixed time window for training to ensure timely block production. Thus, unlike PoW, which assesses miners solely based on speed, PoCL employs a dual-criteria evaluation grounded in both model performance and training time.

- **Single vs. Multiple Winners:** PoW rewards a single miner (or mining pool) per round, reflecting its time-based metric, where only one entity can be the fastest. However, this paradigm is incompatible with decentralized machine learning (DeML), which relies on the aggregation of multiple models to achieve robust global performance. PoCL instead selects the top k performing miners, as determined by the evaluation criteria outlined in the design section. Their

models are aggregated to form the global model, a design choice consistent with standard FL protocols. Additionally, transactions proposed by these k miners can be merged into a single block, which is collectively signed by all selected contributors.

3.3.3 PoCL vs PoFL

Both PoCL and PoFL adapt blockchain consensus mechanisms to incorporate FL in place of traditional proof-of-work (PoW). However, as outlined in the Problem Statement section, PoFL suffers from several key limitations that hinder its practical adoption. This subsection outlines how PoCL addresses these shortcomings, improving efficiency, fairness, and standardization in FL-based consensus.

Collaborative Scope

In PoFL, FL is restricted to individual mining pools, limiting participation to a small subset of miners. This constrained scope reduces both the diversity of data and the overall computational capacity available for training, which in turn hampers global model performance. PoCL removes this restriction by establishing a unified, decentralized FL network in which all participating miners contribute collaboratively. This broader participation enhances the quality and generalizability of the global model.

Model-Sharing Fairness

PoFL's isolated training within pools prevents model sharing across the broader network, disadvantaging newly joined miners who lack access to the more mature models held by long-standing participants. PoCL mitigates this issue by promoting a shared training framework where all miners collectively update and benefit from the same global model, ensuring equitable starting points and fairer competition.

Model Architecture Standardization

PoFL permits each mining pool to define its own machine learning model to solve the requester's task. However, in real-world scenarios, requesters typically specify the architecture they wish to have trained. PoCL reflects this practical requirement by enforcing a globally defined model architecture, which all miners must independently train and validate. This ensures consistency in the training process and aligns better with the needs of requesters.

3.3.4 Deadline Selection

PoCL defines multiple deadlines to regulate the timing of various stages in its consensus mechanism. These include the model proposal deadline, prediction submission deadline, and final voting deadline. This subsection outlines how each of these can be calibrated based on system conditions and operational goals.

Model Proposal Deadline

This deadline specifies the time allocated to miners for training the global model on their local data, starting from the reception of the "start round" signal. As training constitutes the most time-intensive phase of each round, careful calibration is critical. The deadline should consider factors such as model complexity, the volume and quality of local training data, and the desired transaction throughput of the blockchain. The administrator, possibly under the authority of a dedicated committee for preserving decentralization, can dynamically adjust this value based on these factors.

Prediction Submission Deadline

This deadline is intended to restrict miners to using only a legitimate forward pass for generating predictions on the test dataset—thus preserving the integrity of the evaluation process. It should allow sufficient time to perform a forward pass on the global model, but not enough to enable reverse-engineering or label inference. The deadline may be tailored based on the model's computational complexity and the miner's hardware capabilities. In heterogeneous environments, this deadline can be personalized for each miner, enhancing both fairness and security.

Voting Deadline

While less critical to system security, the voting deadline must still be optimized to avoid unnecessary delays. It should provide just enough time for miners to compare predicted and true labels and compute the required evaluation metric (e.g., accuracy or loss). Setting this deadline too high could hinder system throughput, while

too low a value could reduce participation. A balanced value ensures both system responsiveness and architectural stability.

3.3.5 IID vs Non-IID

We assume that the publicly available datasets used to train the requested models are distributed among miners in an Identically and Independently Distributed (IID) manner via an off-chain worker. This setup enhances the efficiency of the global model by mitigating the risks of biased or imbalanced local training data. However, our validation mechanism is also designed to handle non-IID scenarios. Since the evaluation phase focuses on model quality rather than raw performance, the global aggregation of the top k models reduces the risk of overfitting to any single miner's data distribution. By requiring each miner to validate predictions against a robust, locally provided test set, PoCL ensures that the final global model remains generalizable—even when trained on heterogeneous data.

3.4 Experiments & Results

We implemented PoCL using Hyperledger Fabric [22], a modular, permissioned blockchain platform well-suited for enterprise-level distributed applications[1]. The Chaincode-centric architecture of Hyperledger Fabric—where Chaincodes serve as smart contracts—enabled the seamless integration of our consensus mechanism as a dedicated network layer atop the existing infrastructure. We designed and deployed smart contracts for all core operations of our framework, ensuring that no single

[1]https://github.com/tcdt-lab/FL-Validated-Learning

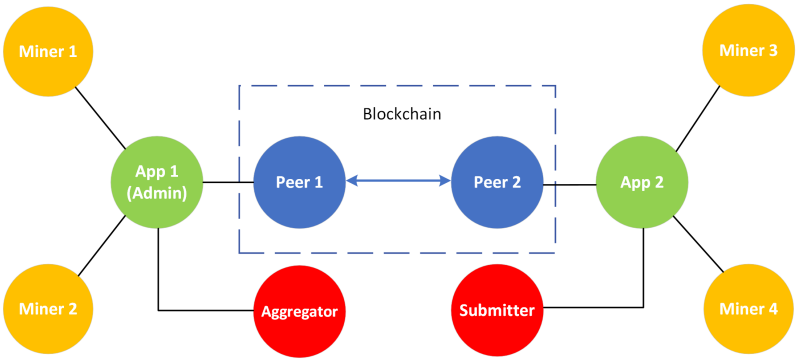


Figure 3.2: PoCL Network architecture

miner or centralized authority can exert unilateral control over the network. Instead, governance is enforced in a decentralized manner through these smart contracts. Figure 3.2 provides a high-level overview of the implemented system, highlighting how the various components of the PoCL framework interconnect.

Our implementation revolves around the functionality of the following elements:

(i) Peers: In Hyperledger Fabric, peers serve as fundamental components of the network, maintaining local copies of the ledger and executing Chaincodes (smart contracts) to endorse transactions. Before appending transactions to the blockchain, peers validate the results against the current ledger state, thereby upholding the system's consistency and integrity. Although this description focuses solely on peer nodes, it is worth noting that orderer nodes—responsible for determining the transaction sequence—are also integral to the Fabric architecture. However, they are omitted here for clarity and brevity.

(ii) Applications: Applications act as intermediaries that facilitate communication with the peers in a Hyperledger Fabric network. They allow for transaction submission and the invocation of Chaincode functions. In our implementation, each peer is

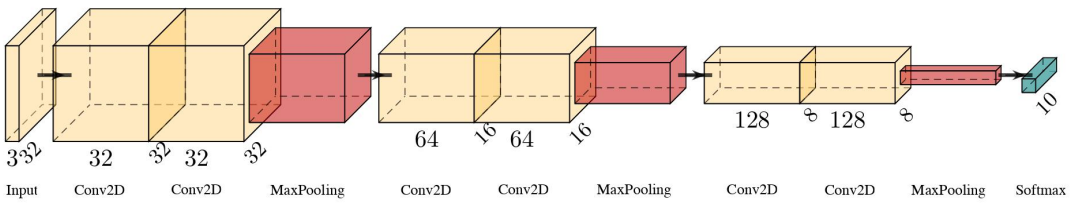


Figure 3.3: PoCL's Evaluation Global Model Architecture.

paired with a corresponding application to streamline interactions. Furthermore, the administrator is realized as one such application to centralize coordination and notification tasks necessary for engaging with miners. Each application is deployed as an Express JS server responsible for handling requests from both miners and end users.

(iii) Miners: Miners are implemented as Flask-based servers that remain idle until prompted by the application to initiate training. In addition to training, miners are responsible for generating predictions on test data and evaluating the predictions submitted by their peers. The training itself is conducted using the TensorFlow framework.

(iv) Aggregator: Similar to the miners, the aggregator is implemented as a Flask server that awaits the command to aggregate updates into a new global model using the FedAvg algorithm. Notably, our design is flexible and can be adapted to incorporate other variations of FL.

(v) Submitter: To realistically emulate user behavior on the blockchain, we developed a program capable of submitting transactions at a configurable rate. Since the blockchain is designed to manage a cryptocurrency system, the transaction submitter generates transfer operations between wallets, simulating real-world fund exchanges.

(vi) Adversaries: Malicious miners may attempt to compromise system integrity by engaging in three specific adversarial behaviors. First, rather than training the global model with their local data, they replace all learned parameters with zeros, intending to degrade the performance of the global model if selected as winners. Second, instead of generating predictions through a legitimate forward pass, they rely on a K-Nearest Neighbors (KNN) algorithm. Lastly, while honest miners rank predictions from most to least accurate during the voting phase, these adversaries intentionally reverse the order, promoting inferior models to increase their chances of success.

We trained a deep Convolutional Neural Network (CNN) [76], illustrated in Fig. 3.3, to perform classification on the CIFAR-10 dataset using ten miners with equal computational capabilities. In every round, the top $K = 5$ miners are selected based on their prediction accuracy and response time, and their models are aggregated through the FedAvg algorithm. To evaluate the fairness of our proposed framework, we deliberately created an imbalanced data distribution by assigning miners six through ten four times more data samples than miners one through five. This configuration was designed to study how varying data volumes influence a miner's likelihood of winning during competitive rounds.

Fig. 3.4 shows the validation loss of each local model over 20 rounds of training. As demonstrated by these plots, all local models show improvement throughout training.

To further investigate the correlation between data size and competitive outcomes, we present the total number of rounds won by each miner in Figure 3.5b. Counter-

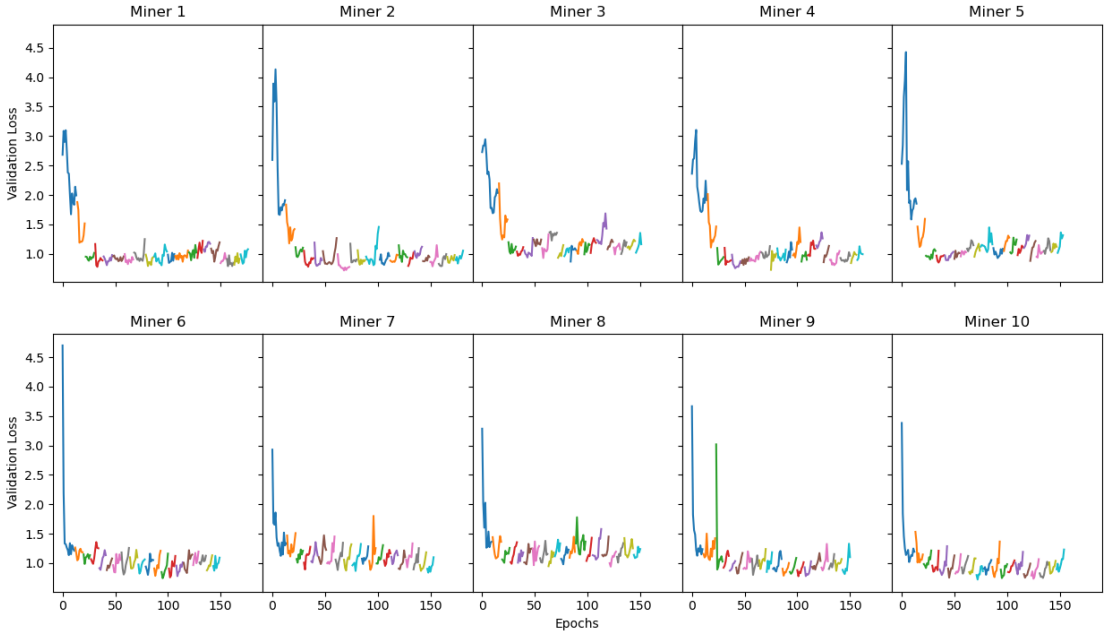
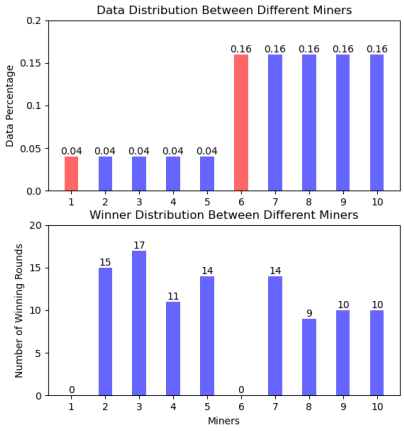


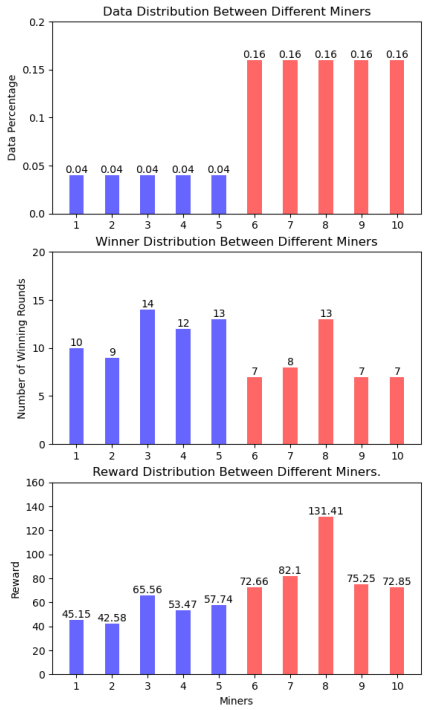
Figure 3.4: PoCL Validation loss through 20 mining rounds. In each subplot, different colors indicate different rounds of training.

intuitively, miners with smaller datasets outperformed their counterparts in terms of win count. This counterintuitive observation prompted a deeper analysis of the round-wise distribution of winners, depicted in Figure 3.6, which reveals that miners with larger datasets tend to dominate the earlier rounds. Moreover, Figure 3.4 demonstrates that their contributions during these early rounds are particularly impactful, as indicated by substantial improvements in validation accuracy and reductions in validation loss.

Our reward mechanism effectively reflects the relative contributions of individual miners, as shown in Figure 3.5b. While miners with smaller datasets may secure more round victories, their overall rewards are lower compared to those with larger datasets. This outcome validates that our framework encourages fair competition on



(a) KNN attack results from adversaries with both small (Miner 1) and large (Miner 6) datasets.



(b) The relationship between data size, rewards, and the number of winning rounds.

Figure 3.5: Comparative performance analysis of PoCL in normal and adversarial settings.

a per-round basis while maintaining a balanced and equitable distribution of rewards across all participating winners.

Additionally, we performed a comparable experiment using identical hyperparameters to evaluate the effect of KNN-based attacks on the system. In this scenario, we introduced two adversarial miners—miner one and miner six, with the latter having four times the amount of data records compared to the former. The results of this experiment, presented in Figure 3.5a, reveal that neither adversary succeeded in winning any competition round, irrespective of their data size.

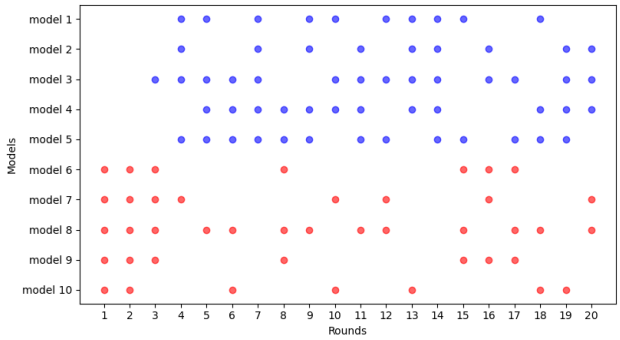


Figure 3.6: PoCL Winning Miners Per Round.

3.5 Conclusion

In this chapter, we introduced Proof-of-Collaborative-Learning (PoCL), a multi-winner consensus mechanism for FL that repurposes the energy efficiency of the traditional proof of work. This framework enables the global training of requested models by utilizing the computational resources of participating miners. To assess miners' performance and identify the winners, we proposed an innovative evaluation step based on the predictions miners make on the test records of their peers. Moreover, we validated the resilience of our evaluation approach by simulating an attack scenario, demonstrating the ineffectiveness of such attacks in disrupting the system. Finally, we introduced a novel reward distribution model that compensates the winning miners based on the significance of their contributions. Through comprehensive experiments, we showed that our reward allocation strategy ensures fair compensation for miners, both within individual rounds and across all rounds.

Chapter 4

Decentralizing SplitFed Learning

4.1 Problem Statement

Protecting data privacy has become a fundamental challenge in the training of machine learning models, particularly as sensitive information is now often spread across decentralized devices. This concern has driven the development of collaborative learning paradigms [27], which enable multiple entities to jointly train a global model without exposing their local data. Instead of exchanging raw datasets, participants share model parameters or gradients, thereby reducing the risk of privacy breaches.

Among the most prominent collaborative learning methodologies are FL [5] and SL [6; 7]. In FL, clients independently train a shared global model using their local data and transmit the resulting model updates to a central server, where techniques such as FedAvg [77] are employed to aggregate these contributions. However, this approach places considerable computational strain on client devices, rendering it less suitable for environments with limited processing power [68].

To alleviate these constraints, SL proposes partitioning the model into two segments: a lightweight client-side portion and a more computationally intensive server-side portion. Clients train the smaller segment locally, while the server processes the remaining layers, thereby easing the computational burden on end devices. Despite this benefit, SL is hindered by slower training times due to its inherently sequential client-server architecture and the need for frequent communication per data batch [27; 28; 29]. Additionally, SL tends to be less efficient than FL, as it does not fully capitalize on aggregation strategies such as FedAvg [8].

To address the limitations of existing approaches, SFL [8] was proposed as a hybrid paradigm that integrates the advantages of both FL and SL. By introducing a federated server responsible for aggregating client-side model updates via the FedAvg algorithm, SFL achieves a notable reduction in training time relative to conventional SL. Nonetheless, as the number of clients increases, the computational load on the SL server intensifies, posing scalability challenges. Moreover, the increase in client participation amplifies communication demands, thereby exerting additional pressure on the network infrastructure.

The scalability challenges associated with both SL and SFL are not limited to infrastructure constraints but also manifest in diminished model performance. Prior studies [65; 66; 67] have shown that increasing the number of clients can exacerbate the disparity between client-side and server-side model updates, ultimately resulting in performance degradation of the global model.

SFL architectures are also vulnerable to critical security threats. For example, a compromised federated server could jeopardize the global model's integrity by selec-

tively incorporating adversarial updates [31]. Likewise, malicious clients may destabilize the training process by contributing corrupted model updates. Furthermore, the reliance on centralized servers introduces single points of failure, which significantly increases operational risks, as these servers retain all contributions to the global model [11].

To address these limitations, we present the first fully decentralized, scalable, blockchain-enabled SplitFed Learning framework. At the core of our design is Sharded SplitFed Learning (SSFL), a novel architecture aimed at improving the scalability and performance of traditional SFL. SSFL distributes the workload of the SL server across multiple parallel shards, with clients assigned to different shards to ensure balanced computational demands and improved training efficiency. To resolve the issue of imbalanced model updates inherent in SFL, SSFL introduces an additional federated server that aggregates the models from each shard using the FedAvg algorithm [5]. This aggregation mechanism effectively stabilizes the training process by smoothing shard-specific model updates and mitigating the detrimental effects of elevated local learning rates and server-induced biases. As a result, SSFL achieves greater scalability while also enhancing model convergence and training stability.

Building on the SSFL architecture, we further enhance our design by integrating blockchain technology to address ongoing security concerns stemming from centralized control [78]. We introduce Blockchain-enabled SplitFed Learning (BSFL), the first decentralized SFL framework that removes the dependency on a central FL server and strengthens security through a committee-driven blockchain consensus protocol [31]. In BSFL, core server responsibilities—including model aggregation, update val-

idation, and reward allocation—are delegated to smart contracts deployed on the blockchain. A decentralized committee of validator nodes assesses model updates from each SSFL shard, thereby mitigating risks such as data poisoning and preserving the integrity of collaborative contributions. This blockchain-based approach establishes a secure, trustless environment that ensures fairness, transparency, and tamper resistance through decentralized consensus.

We summarise the contributions of this chapter as:

1. Sharded SplitFed Learning (SSFL): We introduce SSFL as an advanced version of the existing SFL model, designed to address its scalability challenges. This approach alleviates the computational burden on the SL server by enabling parallel training across multiple SFL shards, significantly improving both the scalability and efficiency of the model. Additionally, SSFL tackles the issue of imbalanced model updates by reducing the effective learning rate through the use of federated averaging. SSFL proves particularly advantageous in environments with a large number of resource-constrained devices, where efficient training and model convergence are crucial.

2. We present BSFL, the first blockchain-powered SFL framework, designed to resolve the security vulnerabilities arising from the centralization of the traditional SFL approach. BSFL replaces the central FL server with a decentralized blockchain system, utilizing a committee-based consensus mechanism to ensure robust model aggregation and validation. This architecture incorporates an evaluation process for client model updates, effectively mitigating risks such as data poisoning and model tampering, while simultaneously enhancing the

system's fairness, robustness, and overall performance.

3. To validate the effectiveness of SSFL and BSFL, we conduct extensive experiments assessing their performance under both standard operating conditions and in the presence of simulated data-poisoning attacks. Furthermore, we compare the round completion time and convergence speed of our proposed frameworks against traditional SL and SFL models, demonstrating that SSFL and BSFL deliver superior performance and operational efficiency.

4.2 Sharded SplitFed Learning

As a variation of SL, SFL merges the advantages of both SL and FL to address the limitations inherent in each approach. The SFL architecture introduces two key innovations: parallel client training and a federated server that aggregates the client models at each training round. While these modifications substantially reduce the convergence time of SL, scalability remains a critical challenge. In both SFL and SL, as the number of clients increases, the computational and communication overhead on the central SL server grows significantly, which limits the applicability of these architectures in large-scale, real-world scenarios.

However, the scalability challenges in SL and its variations are not solely driven by infrastructure capacity. Both SL and SFL experience performance degradation that worsens as the number of clients increases. In these frameworks, the server-side model is updated far more frequently than the client-side model on each local device, creating an imbalance in the training process. As a result, this disparity between the

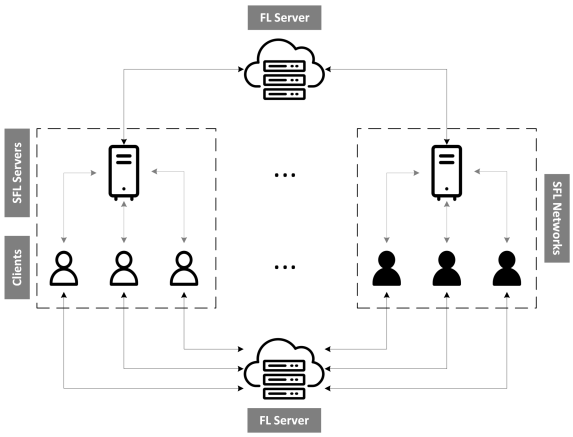


Figure 4.1: Overview of Sharded SplitFed Learning framework.

server and client models becomes more pronounced as the client count grows, leading to a significant decline in overall performance. While previous studies have proposed various solutions to address the performance and efficiency limitations of SL and SFL in scalable environments [65; 66; 67], to the best of our knowledge, no work has fully addressed the imbalanced training problem, nor the challenges related to server-side computation and communication overhead.

To address these challenges, we propose Sharded SplitFed Learning (SSFL), an enhanced framework in which clients are organized across multiple shard servers, with each shard functioning as a conventional SL server. As depicted in Figure 4.1, the SSFL architecture introduces an additional federated server responsible for aggregating model updates from all shard servers in parallel. Although we represent the two FL servers as separate entities, their roles could be consolidated into a single instance if specific security measures, such as encrypted model updates between clients and servers, are implemented. For the sake of simplicity in our explanation, we assume these security precautions are in place, and the system consists of a single FL server

Table 4.1: SSFL and BSFL Notations

Symbol	Description
R	Total number of training rounds in a shard server
T	Total number of training cycles
E	Number of epochs for each client's local training
J	Number of clients per shard
I	Total number of shard servers in the SSFL network
K	Number of top committee-selected model updates in BSFL
$W^S_{i,j,r}$	Model weights of shard server i for client j at round r
$W^C_{j,r}$	Local model weights of client j at round r
W^S_t	Server global model at cycle t
W^C_t	Client global model at cycle t
X_j	Input data for client j
Y_j	True labels associated with X_j
$A_{j,r}$	Activation output from client j at round r
$\hat{Y}_{j,r}$	Predicted labels for client j at round r
$\mathcal{L}_{j,r}$	Loss computed for client j at round r
$\nabla \ell_j(W^S_{i,j,r}, A_{j,r})$	Gradient of the loss function with respect to $W^S_{i,j,r}$ and $A_{j,r}$
$dA_{j,r}$	Feedback gradient (from shard server to client) for client j at round r
$\nabla \ell_j(W^C_{j,r})$	Gradient of the loss function with respect to the local model weights $W^C_{j,r}$ for client j at round r
λ	Learning rate for updating local client models

that aggregates both client and SL server models in the SFL framework.

4.2.1 SSFL Workflow

To initiate training, clients join an SFL network by sending a handshake request to one of the available SFL servers. Once connected, clients begin the training process

Algorithm 2 Sharded SplitFed Learning (SSFL)

/ Executes on each Shard Server (SFL Server) */*

Ensure: **TrainingCycle**(c):

for each round $r = 1, 2, \ldots, R$ **do**

  for each client $j \in$ shard in parallel **do**

    for each batch b in epoch E **do**

      $(A_{j,r}, Y_{j,r}) \leftarrow \text{ClientTrain}(W^C_{j,r})$

      $\hat{Y}_{j,r} \leftarrow \text{ServerForwardPass}(A_{j,r}, W^S_{i,j,r})$

      $\mathcal{L}_{j,r} \leftarrow \text{Loss}(Y_{j,r}, \hat{Y}_{j,r})$

      $\nabla \ell_j(W^S_{i,j,r}, A_{j,r}) \leftarrow \text{ComputeGradients}(\mathcal{L}_{j,r})$

      $W^S_{i,j,r} \leftarrow W^S_{i,j,r} - \lambda \cdot \nabla \ell_j(W^S_{i,j,r})$

      Send $dA_{j,r} := \nabla \ell_j(W^S_{i,j,r}, A_{j,r})$ to client j.

    end for

  end for

  Update shard server model:

  $W^S_{i,r+1} \leftarrow \frac{1}{J} \sum_{j=1}^{J} W^S_{i,j,r}$

end for

/ Executes on the FL Server */*

Ensure: **Training**():

for each training cycle $t = 1, 2, \ldots, T$ **do**

  for each shard server $i \in$ SFL network in parallel **do**

    Call **TrainingCycle**(t) for shard i.

  end for

end for

22: */* Executes on the FL Server */*

Ensure: Aggregate():

23: Receive $W_{i,t}^S$ and $W_{j,t}^C$ updates from each shard server i and client j.

24: Update global models:

25: $W_{t+1}^S \leftarrow \frac{1}{I} \sum_{i=1}^{I} W_{i,t}^S$

26: $W_{t+1}^C \leftarrow \frac{1}{J} \sum_{j=1}^{J} W_{j,t}^C$

by performing a forward pass on the global client model using their local data. The outputs and targets for each batch are sent to the designated SFL server, which continues the training by performing a forward pass through the split layers. Upon reaching the final layer, the server computes the gradients and updates the local SFL server model. These gradients are then sent back to the clients, enabling them to update their respective models. After all shards complete local training and aggregation, both the clients and SFL servers send their updated models to the FL server for a final aggregation step. Each FL aggregation completes a cycle of training in SSFL. To formalize this process, we present Algorithms 0 and 0, which outline the detailed workflow of each component in sequence. Additionally, Table 4.1 provides a comprehensive notation reference that defines the variables, parameters, and functions used in these algorithms, enhancing clarity and aiding understanding.

4.2.2 Scalability and Performance

Incorporating shards into the SFL network significantly alleviates the computational burden on individual SFL servers by distributing the forward and backward

Algorithm 3 Client Training in SplitFed Learning

/* *Executes on each client device* */

Ensure: ClientTrain():

for each batch b in epoch E **do**

  $A_{j,r} \leftarrow \text{ClientForwardPass}(X_j, W^C_{j,r})$

  Assuming Y_j is the label of X_j

  Send $(A_{j,r}, Y_j)$ to the assigned shard server

end for

/* *Executes on each client after receiving the gradients* */

Ensure: ClientBackProp():

Receive $dA_{j,r}$ from shard server.

$\nabla \ell_j(W^C_{j,r}) \leftarrow \text{BackPropagate}(dA_{j,r})$

$W^C_{j,r} \leftarrow W^C_{j,r} - \lambda \cdot \nabla \ell_j(W^C_{j,r})$

propagation tasks across multiple servers. This sharding approach not only improves scalability by effectively expanding the system's infrastructure capacity but also enhances performance as the number of clients increases. Additionally, the sharding mechanism addresses the scalability challenges related to performance degradation, which arise from the imbalance in learning rates between the SFL server and client models.

As discussed by [65], the performance issue arises from the SFL server's effective learning rate being higher than that of the clients, causing updates to the SFL server model to occur much faster than those to the client models. This discrepancy leads to

unstable and suboptimal training, particularly as the number of clients increases. To mitigate this imbalance, studies like LocFedMix [65] and SGLR [66] have proposed solutions, including augmenting the client data or adjusting the learning rate to better synchronize client and server updates.

In contrast, our approach introduces an additional federated server to aggregate updates from the shard SFL servers. This extra FL server layer helps reduce the learning rate of the shard servers, creating a more balanced and synchronized training process. This adjustment not only mitigates the risk of performance degradation but also ensures that both the SFL server and client models progress at a harmonious rate, thereby improving the overall stability and efficiency of the training framework.

4.3 Blockchain-enabled SplitFed Learning

While sharding enhances both the performance and scalability of the SSFL framework, centralization at the server side introduces several security and reliability risks: (i) The FL server may exhibit bias, either intentionally or due to external manipulation, which can disrupt the training loop and undermine both model integrity and overall training efficiency. (ii) The central FL server in SSFL and its parent systems handle all core functionalities and bear the main workload, making it a critical single point of failure. Any failure of the FL server could halt the system's operation and risk the loss or corruption of all contributions, significantly impacting system reliability and continuity. (iii) Malicious clients in the SSFL framework may engage in data poisoning attacks, submitting manipulated data to compromise the quality, reliability, and generalizability of the global model.

Algorithm 4 Blockchain-enabled SplitFed Learning (BSFL)

Ensure: TrainingCycle(c):

for each round $r = 1, 2, \ldots, R$ **do**
  for each client $j \in$ shard in parallel **do**
    for each batch b in epoch E **do**
      $(A_{j,r}, Y_{j,r}) \leftarrow \text{ClientTrain}(W^C_{j,r})$
      $\hat{Y}_{j,r} \leftarrow \text{ServerForwardPass}(A_{j,r}, W^S_{i,j,r})$
      $\mathcal{L}_{j,r} \leftarrow \text{Loss}(Y_{j,r}, \hat{Y}_{j,r})$
      $\nabla \ell_j(W^S_{i,j,r}, A_{j,r}) \leftarrow \text{ComputeGradients}(\mathcal{L}_{j,r})$
      Send $dA_{j,r} := \nabla \ell_j(W^S_{i,j,r}, A_{j,r})$ to each client j.
    end for
  end for
  Update shard server model:
  $W^S_{i,r+1} \leftarrow \frac{1}{J} \sum_{j=1}^{J} W^S_{i,j,r}$
end for
Send $W^S_{i,R}$ to blockchain ledger.
Request each client $j \in$ shard to submit $W^C_{j,R}$ to ledger.

Ensure: Evaluate($W^S_{i,R}$, [$W^C_{j,R}$ for each client j]):

Initialize **scores** as an empty list.
for each client $j \in$ shard in parallel **do**
  $A_{j,R} \leftarrow \text{ClientForwardPass}(X, W^C_{j,R})$
  $\hat{Y}_{j,R} \leftarrow \text{ServerForwardPass}(A_{j,R}, W^S_{i,R})$
  $\mathcal{L}_{j,R} \leftarrow \text{Loss}(Y_{j,R}, \hat{Y}_{j,R})$
  Append $\mathcal{L}_{j,R}$ (validation loss) to **scores**.
end for
Return Median(**scores**)

24: */* Runs on the blockchain network */*

Ensure: Committee Procedure:

25: **for** each cycle $t = 1, 2, \ldots, T$ **do**

26: **if** $t = 1$ **then**

27: Randomly select committee members.

28: **else**

29: Select committee members based on scores from the previous cycle.

30: **end if**

31: **for** each committee member i in parallel **do**

32: **TrainingCycle**(t)

33: **end for**

34: Receive all $W^S_{i,R}$ and $[W^C_{j,R}]$ updates.

35: **for** each member i **do**

36: Send $W^S_{i,R}$ and $[W^C_{j,R}]$ to other committee members.

37: Call **Evaluate** to obtain validation scores.

38: **end for**

39: Assign the median of all scores given to shard i as the final score.

40: Sort scores and select top k model updates.

41: Update global models:

42: $W^S_{t+1} \leftarrow \frac{1}{K} \sum_{k=1}^{K} W^S_{k,t}$

43: $W^C_{t+1} \leftarrow \frac{1}{K \cdot J} \sum_{k=1}^{K} \sum_{j=1}^{J} W^C_{k,j,t}$

44: **end for**

(iv) Centralization increases the risk of privacy breaches, as the FL server manages

sensitive information from all clients. A breach or unauthorized access to the central server could lead to the exposure of confidential client data, violating core privacy and confidentiality requirements. (v) The framework inherently assumes that the central FL server is fully trustworthy and operates without malicious intent. However, if the server is compromised or behaves maliciously, such as by colluding with external entities, it could jeopardize the system's security, fairness, and overall trustworthiness.

To mitigate the aforementioned challenges, we propose decentralizing the SSFL framework using blockchain technology, thereby eliminating the reliance on a centralized FL server and its associated security vulnerabilities. In our Blockchain-enabled SplitFed Learning (BSFL) architecture, smart contracts are leveraged to assume the responsibilities traditionally held by the central server, enabling fully decentralized, end-to-end model training. We further introduce a committee-based consensus mechanism within BSFL that governs block creation, model update evaluation, and validation, thereby ensuring secure, impartial, and robust system operation. Through the combined design of SSFL and its decentralization via BSFL, this work presents the first Blockchain-enabled SplitFed Learning system, significantly advancing the fairness, efficiency, and scalability of the original SFL framework.

To decentralize the responsibilities of the FL server, we initialize both the global client and SFL server models directly on the blockchain. This configuration enables clients and SFL servers to retrieve the global models seamlessly and initiate the training process. After a predefined number of SFL training rounds, each client and SFL server submits their locally updated models to the blockchain. This submission triggers an aggregation smart contract, which autonomously combines the updates

into a new global model, thereby ensuring decentralized, secure, and transparent model evolution.

4.3.1 Committee Consensus Mechanism

Consensus mechanisms in blockchains determine the content of blocks and their order within the chain. In this paper, we employ a committee consensus mechanism to evaluate model updates proposed by each shard, selecting only the top-performing models for aggregation. In BSFL, the SFL servers in each shard form a committee that validates each other's model updates by assigning a score to each update during each training cycle. By limiting communication to the committee nodes, the system significantly reduces communication overhead. The scores assigned to committee members are then ranked, and only the top K models are aggregated to create the updated global models. At the start of each new cycle, a new committee is selected based on the scores from the previous cycle, ensuring that prior committee members do not serve consecutively, therefore reducing the risk of collusion or malicious actions.

In blockchain systems, consensus mechanisms are essential for determining both the contents and sequencing of blocks. Within our proposed BSFL framework, we implement a committee-based consensus mechanism to evaluate model updates submitted by each shard, selectively aggregating only the highest-quality contributions. Specifically, SFL servers within each shard constitute a committee that assesses the model updates of their peers during each training cycle by assigning performance scores. This selective, intra-committee communication strategy effectively reduces network overhead. Following evaluation, updates are ranked based on these scores,

and only the top K models are aggregated to form the new global models. To enhance fairness and security, a new committee is elected at the beginning of each training cycle, based on the scoring outcomes from the previous round. This rotation mechanism prevents consecutive participation of the same nodes, thereby mitigating the risk of collusion or malicious behavior.

Each committee member validates model updates proposed by other members by evaluating them on their own local dataset. The performance metric, such as validation loss or accuracy, is used to quantify the quality of each model update. This value is then reported as the score assigned by one committee member to another. To ensure robustness against outliers or biased evaluations, the final score for each committee member in a given cycle is computed as the median of all received scores.

The evaluation metric used in scoring can be either validation loss or validation accuracy, depending on the nature of the task. The key distinction between the two lies in the optimization objective: validation loss is minimized, whereas validation accuracy is maximized. Moreover, validation accuracy is only applicable to classification tasks, while validation loss serves as a more general-purpose metric that can be employed across a wider range of machine learning problems, including regression and other non-classification scenarios.

4.3.2 BSFL Workflow

BSFL supervises the training procedure through the use of three distinct smart contracts, each replicating specific responsibilities of the central FL server. In the initial round, the *AssignNodes* smart contract randomly selects nodes within the network

to act as SFL servers for each shard and assigns clients to these servers, thereby defining the shard composition. Each shard then proceeds with standard SFL training. Upon completion, the resulting models are stored on the blockchain. The *ModelPropose* smart contract is responsible for collecting all trained models and distributing them to every committee member, i.e., the SFL servers. Each server evaluates the submitted models using its local validation data and assigns a validation loss score to the models of other members. The final score for each model is determined as the median of the scores it receives. These results are passed to the *EvaluationPropose* smart contract, which ranks the models and selects the top K performers. The aggregate of these top K models forms the updated global models for the next training cycle. The complete BSFL training workflow is detailed in Algorithms 4 and 0.

4.3.3 Node Assignment

The selection of the committee for each training cycle is governed by the performance scores of nodes from the preceding round. To uphold fairness and enhance the system's security, nodes that participated as committee members in round t are barred from committee participation in round $t + 1$. A node is appointed as an SFL server if its score exceeds that of other nodes not involved in the previous committee. Shards are then constructed sequentially: each chosen server is assigned a subset of clients to constitute a shard, and this process is iteratively applied for the remaining shards. Notably, nodes that served as committee members in the prior cycle are exclusively eligible for client roles in the current cycle, ensuring role rotation and reducing the potential for collusion.

The method used to assign nodes as clients and SFL servers could differ from the one we propose. Our approach organizes nodes with comparable efficiency into the same shard, ensuring that even if data-poisoning attacks occur, the performance of the honest nodes is less likely to be compromised or overshadowed.

4.4 Discussions

4.4.1 Efficiency

The BSFL framework improves efficiency by aggregating only the highest-performing models, leading to a more refined global model than what is achieved in SSFL. Furthermore, the rotating selection of committee members guarantees that the global model is trained on all local datasets throughout the system, effectively performing a distributed form of k-fold cross-validation over time.

4.4.2 Security and Stability

By substituting the centralized FL server with a blockchain network, the system's reliability is significantly improved, as the central FL server no longer represents a single point of failure. In this blockchain-driven framework, all operations are executed through smart contracts and validated by a committee of nodes. This decentralized approach to validation ensures the training process remains secure and continuous, reducing the risks typically associated with central FL servers.

4.4.3 Non-IID Data

A recurring issue in distributed learning is managing Non-Independent and Identically Distributed (Non-IID) local datasets. In such settings, each device may possess a skewed or unbalanced dataset—for example, one device may predominantly contain images of cats, while another may primarily include images of dogs. This imbalance results in local data distributions that diverge significantly from the global distribution, thereby complicating model convergence and generalization.

To address this challenge, our framework leverages data from committee members for cross-validation, thereby exposing the global model to a broader and more diverse range of data during the evaluation phase. This approach helps mitigate the risk of overfitting by reducing the model's reliance on the specific, and potentially skewed, data distributions of individual nodes. By validating model updates against varied committee datasets, the system promotes the development of a more robust and generalized global model, better suited to handle the heterogeneity inherent in non-IID data.

4.4.4 Committee Election

In BSFL, we propose the selection of committee members based on the scores of clients from the previous round. Alternatively, a random selection process could be utilized to promote model generalization by ensuring the global model is exposed to a broader array of datasets over multiple training cycles.

While our current approach maximizes performance, stability, and fairness by selecting committee members from all clients in the prior round, this could seem

Table 4.2: BSFL's Evaluation Client and Server Model Architectures.
$(D \times H \times W)$ are the dimensions of input data.

Model	**Layer Type**	**Parameters**	**Output Shape**
Client Model	Conv2d	In: D, Out: 32, Kernel: 3×3	$32 \times H \times W$
	ReLU	-	$32 \times 28 \times 28$
	MaxPool2d	Kernel: 2×2, Stride: 2	$16 \times H/2 \times W/2$
Server Model	Conv2d	In: 32, Out: 64, Kernel: 3×3	$64 \times H/2 \times W/2$
	ReLU	-	$64 \times 14 \times 14$
	MaxPool2d	Kernel: 2×2, Stride: 2	$64 \times H/4 \times W/4$
	Flatten	-	$1 \times (64 \times H/4 \times W/4)$
	Linear	In: $(64 \times H/4 \times W/4)$, Out: 128	1×128
	ReLU	-	1×128
	Linear	In: 128, Out: 10	1×10

at odds with the objective of SL, which aims to reduce the computational load on resource-limited clients. To address this concern, a more refined committee selection strategy could be implemented, factoring in the computational capabilities of nodes. Limiting committee membership to nodes capable of managing shard server tasks ensures that only appropriate nodes are selected, thereby preserving the integrity of SL while also promoting fairness and efficiency.

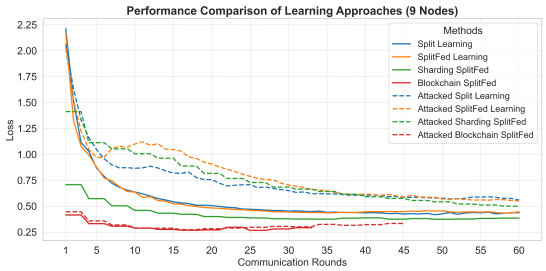
4.4.5 Malicious Nodes

In SSFL, the impact of malicious nodes on system performance varies based on their assigned roles. As clients, such nodes may submit harmful model updates, thereby degrading the accuracy of the global model. When functioning as FL servers,

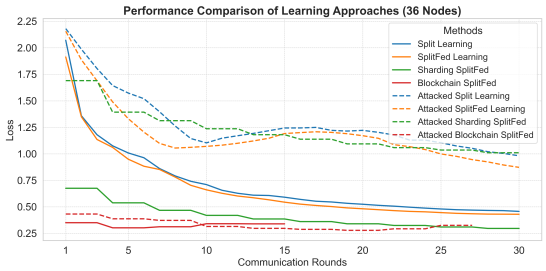
they can inject noise into the aggregated model or exhibit biased behavior by selectively favoring specific clients, ultimately reducing the model's generalizability. To mitigate these risks, we propose the integration of a blockchain network with a committee-based consensus mechanism. This approach eliminates reliance on a central FL server and provides a robust framework for evaluating model updates. The following sections describe how BSFL enhances resilience against malicious nodes in two critical scenarios:

1. **Malicious Clients:** Malicious nodes may submit "poisonous" updates that degrade the performance of the global model. Our proposed evaluation and committee selection algorithm mitigates these threats by filtering out harmful updates and retaining only the top K most reliable model submissions. As long as there are at least $K \times C$ honest clients in the system (where C denotes the number of clients per shard), the system can continue to train effectively and resist major disruptions from data-poisoning attacks.

2. **Malicious Committee Members**: Malicious nodes may infiltrate the committee and attempt to disrupt the evaluation process. In a committee consisting of N nodes, each model update is evaluated by the other $N-1$ members, with the final score being determined as the median through a smart contract. While malicious members may try to skew the results in favor of subpar updates, their influence is limited unless they form the majority. To maintain evaluation integrity, the BSFL framework requires that at least $\lfloor \frac{N}{2} \rfloor + 1$ committee members be honest. Additionally, to prevent the aggregation of harmful updates, the number of selected models, K, must be kept under $\frac{N}{2}$.



(a) Performance comparison for 9 nodes in 60 training rounds.



(b) Performance comparison for 36 nodes in 30 training rounds.

Figure 4.2: SL, SFL, SSFL, and BSFL Validation Loss Comparison For Distinct Network Sizes.

In conclusion, robust security in BSFL relies on having $\lfloor \frac{N}{2} \rfloor + 1$ honest committee members, at least $K \times C$ honest clients, and selecting K values that satisfy the condition $2 < K < \frac{N}{2}$. However, BSFL's security parameters are flexible: in cases of minimal malicious activity, the value of K can be adjusted to prioritize efficiency while still maintaining adequate security.

4.5 Results

To assess the effectiveness of the SSFL and BSFL frameworks, we implemented both models using the Fashion MNIST dataset, which contains 60,000 images categorized into 10 distinct classes.[1] In this study, a node refers to any participant in the distributed learning process, whether functioning as a client or as an SFL server. This terminology is particularly relevant in the BSFL framework, where roles are dynamic

[1]The corresponding implementation is available in the following link: **https://github.com/icdcs249/SSFL-BSFL**

and rotate across training cycles. Experiments were conducted under two configurations: one with 9 nodes and another with 36 nodes, allowing for a comprehensive assessment of the frameworks under both small-scale and large-scale deployment scenarios. Each node received an equally sized subset of 6,666 images; however, the data distributions were deliberately constructed to be non-IID to emulate real-world heterogeneity.

Approaches	**Normal Test Loss**	**Attacked Test Loss**	**Avg. Round Time (min)**
Split Learning (SL)	0.456	0.981	37.6
SplitFed Learning (SFL)	0.430	0.872	37.2
Sharding SplitFed Learning (SSFL)	0.296	1.010	5.5
Blockchain-enabled SplitFed Learning (BSFL)	0.339	0.325	33.7

Table 4.3: Performance comparison of learning approaches: normal and attacked test losses and average training times per round (36 nodes)

Furthermore, the BSFL architecture was implemented using the Hyperledger Fabric framework to support the execution of smart contracts (also referred to as chaincodes). Each node in our system integrates PyTorch for machine learning tasks and Flask to facilitate communication between the nodes and the underlying blockchain infrastructure. In our experimental environment, each node operates as an independent process. All experiments were conducted on a system equipped with an Intel Xeon(R) 4216 processor, comprising 16 physical cores and 32 logical threads (two threads per core), and accelerated by an NVIDIA GeForce RTX 3080 GPU to support efficient model training.

To ensure a fair comparison, we evaluate the performance of both SSFL and BSFL

under identical experimental conditions alongside traditional SFL and SL frameworks. In each setup, the objective is to train global models—outlined in Table 4.2—to perform image classification on the Fashion MNIST dataset. To maintain consistency and enable an accurate assessment of each approach, all experiments are conducted using the same set of fixed hyperparameters.

In the SL and SFL configurations, a single node—either one out of nine or one out of thirty-six—acts as the central server responsible for managing the training process. For the SSFL and BSFL experiments, we partition the system into three shards, each comprising two clients, when operating with nine nodes. In the 36-node setup, the system is restructured into six shards, each consisting of five clients. The parameter K, which dictates how many of the highest-performing shard SFL servers contribute to the global model aggregation, is set to two in the smaller setup and three in the larger configuration.

Early stopping is applied uniformly across all frameworks to mitigate the risk of overfitting. This technique is easily incorporated into SSFL, SFL, and SL, where a central coordinating node monitors the training progress. In the BSFL setting, early stopping is governed by the committee consensus mechanism, which terminates training once a rise in validation loss signals the onset of overfitting.

Figures 4.2a and 4.2b show the validation loss of SL, SFL, SSFL, and BSFL across all training rounds. As evident in both figures, SSFL and BSFL outperform the earlier approaches, primarily due to the sharding mechanism employed in these frameworks. In both frameworks, the sharded architecture eradicates the imbalanced learning rate of SL by aggregating the SFL server models, enhancing global model

performance. Furthermore, BSFL demonstrates superior performance compared to SSFL by utilizing its committee consensus evaluation, which ensures that only the top-performing model updates contribute to the global model. Notably, BSFL requires fewer training rounds, as its enhanced performance leads the global model to reach optimal accuracy and begin overfitting the training data more quickly. Finally, Table 4.3 compares the final test loss of all distributed learning algorithms. As shown, SSFL substantially enhances scalability and performance over SL and SFL due to its lower round time and test loss. Nevertheless, the test loss in the attacked settings proves SSFL's inability to tolerate data-poisoning threats.

4.5.1 Under Malicious Attacks

In our experiments, we assume that malicious nodes carry out data poisoning attacks by sending harmful updates to the central FL server, thereby undermining the performance of the global model. We conduct these attacks across all approaches to validate the effectiveness of our proposed solution. To evaluate the impact of data poisoning attacks, we test with varying proportions of malicious nodes, specifically 33% and 47% for the 9-node and 36-node setups, respectively. These varied attacker proportions allow us to evaluate the resilience and performance of our proposed frameworks under different threat levels. Notably, the 36-node experiment with 47% attackers represents a scenario where the system is pushed to its limits, simulating the maximum number of attackers possible without breaching the 51% threshold typically required for a successful blockchain takeover.

In BSFL, we further simulate a voting attack to assess the resilience of the com-

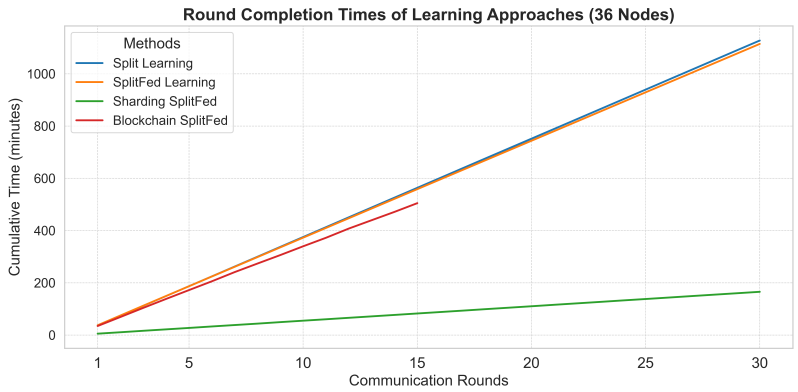


Figure 4.3: BSFL Round Completion Times For 36 Nodes.

mittee consensus mechanism. In this scenario, malicious nodes, when selected as committee members, deliberately vote for the worst-performing model updates to disrupt the aggregation process. By examining the system's ability to counter this disruption, we evaluate whether BSFL can still effectively aggregate the most beneficial model updates.

Figures 4.2a and 4.2b illustrate the performance of each algorithm under identical data-poisoning attacks. As shown in these figures, all algorithms except BSFL are adversely impacted by these attacks, lacking effective mechanisms to counteract them. In SL, SFL, and SSFL, data-poisoning attacks cause a substantial increase in validation loss and decline in the global model's performance. BSFL, however, remains entirely unaffected due to its robust committee consensus mechanism, which effectively filters out malicious updates from adversarial nodes. The effectiveness of BSFL in mitigating data-poisoning attacks is also demonstrated by the test loss values presented in Table 4.3.

4.5.2 Round Completion Time

SSFL reduces transmission costs and communication overhead in SFL training by employing parallel shards, which distribute the workload across multiple nodes. BSFL incorporates blockchain technology to enhance the security of the system, which, in turn, increases the communication load due to committee member coordination and propagation of updates to and from the blockchain. In our experiments, we evaluate the round completion time of all training frameworks by measuring the time between the start and the end of each round and cycle, calculating the computation as well as the communication overhead for each round. This evaluation helps with identifying the most suitable distributed learning method by considering specific constraints, such as time, infrastructure, and other critical factors.

Figures 4.2a and 4.2b present the validation loss trends for SL, SFL, SSFL, and BSFL over successive training rounds. It is evident from both figures that SSFL and BSFL consistently outperform the traditional models, primarily due to the implementation of a sharding strategy. This design addresses the uneven training progression often encountered in SL by aggregating updates from multiple SFL servers, thereby enhancing the quality of the global model. Among the two, BSFL achieves even better results by leveraging a committee-based consensus evaluation, which ensures that only the most effective model updates influence the global model. Importantly, BSFL reaches optimal performance in fewer training rounds, as its improved learning dynamics lead to faster convergence and earlier onset of overfitting. Table 4.3 summarizes the final test losses across all approaches. While SSFL demonstrates improved scalability and performance compared to SL and SFL, attributable to reduced

round durations and lower test losses, it remains vulnerable in adversarial settings, as indicated by the elevated test losses observed under attack scenarios.

4.6 Conclusion

This chapter proposed two advanced extensions to the SFL paradigm: SSFL and BSFL. SSFL addresses the scalability limitations of conventional SFL by distributing the server-side responsibilities across multiple parallel shards. In contrast, BSFL enhances system robustness and security by removing the centralized server and integrating a decentralized blockchain infrastructure. This infrastructure employs a committee-based consensus protocol and a structured evaluation mechanism to protect against the incorporation of malicious model updates. The effectiveness of both frameworks was thoroughly evaluated through a series of experiments, comparing their performance with standard SFL approaches across a range of scenarios. Our empirical results demonstrate significant improvements in scalability, resilience, and training efficiency achieved through the adoption of SSFL and BSFL.

Chapter 5

Fair Model Ownership in Federated Learning

5.1 Problem Statement

In the era of unprecedented Artificial Intelligence (AI) adaptation, data privacy has become one of the most crucial concerns of individuals, organizations, and governments. Consequently, collaborative learning schemes, including FL [5] and SL [6; 7], have become increasingly popular by providing an environment where machine learning (ML) training takes place without the need for data sharing.

FL refers to a variety of distributed learning algorithms where clients train a global ML model by sharing model updates instead of private data records. Although FL introduces a groundbreaking opportunity for privacy-aware ML, it lacks essential fairness, robustness, and transparency issues.

Let us assess a scenario where a few enterprise-level organizations are considering

participating in a FL procedure to collectively train a model without sharing data, a task that would benefit all parties by producing a well-trained model. Nevertheless, many deficiencies of FedAvg prevent such adaptation:

- **Model Ownership**: FL lacks an essential consideration in model ownership. Inherently, the contributions of participating organizations are not alike; nevertheless, their access to the global model is not restricted relative to their contributions. This means that an organization can enjoy all the benefits of the global model while making limited contributions to the training step (free-rider problem) [79; 80].

- **Centralization**: FL's centralized architecture produces stability, security, and robustness concerns. In traditional FL, the aggregator becomes a single point of failure, where, in the case of its collapse, a significant portion of training progress becomes lost [11]. In addition, the central server is a black-box component in the architecture, and its functionalities are not transparent to all clients and organizations.

For these reasons, the adaptation of FL between enterprises is fairly limited, since any risk in fairness, stability, or the security of the system might bring about significant and irreversible damages. To counteract these obstacles, we propose Blockchain-enabled Personalized Federated Learning (BPFL), a novel decentralized FL framework that enhances fairness and robustness of traditional FedAvg. In this design, we introduce a contribution metric computed for each client as a composite value combining global and local loss improvement. We aggregate local models into a new global model using weights derived from the softmax of client contributions, effectively

normalizing contribution scores across all participants. Furthermore, we resolve the fairness issue of FedAvg by personalizing each local client as the interpolation between the submitted local model and the newly computed global model relative to the normalized contribution value. In other words, we generate personalized models for each client according to its contribution value: greater contributions receive models closer to the new global model, and more inferior contributions receive a hardly improved version of their submitted model. In addition, we introduce a tokenized model access, where organizations that are unable to make significant contributions can purchase the global model through the exchange of tokens. At last, we introduce a novel reward mechanism for training nodes where we distribute a constant number of tokens between participants proportional to their training contributions. We decentralize our design by substituting the centralized server with a blockchain network, a well-justified choice due to our tokenization scheme and model commerce choice. The integration of our personalized FL scheme with blockchain is further justified due to its ability to reinforce the security, stability, and transparency of FL.

To summarize, the contributions of this chapter are as follows:

- We define a novel composite contribution metric that integrates global loss reduction on a shared synthetic test set with each client's local training progress, enabling a fair measurement of individual utility.

- We design a softmax-weighted aggregation mechanism coupled with a fairness-driven personalization update, whereby each client's received model is interpolated between its own update and the global model in proportion to its normalized contribution.

- We introduce a tokenized model access scheme—also referred to as model acquisition—allowing clients who cannot contribute sufficiently to acquire the global model via on-chain token exchange.
- We propose a reward distribution protocol that allocates a fixed pool of tokens among participants based on their measured contributions, fostering sustained engagement and incentive alignment.
- We replace the centralized aggregator with a permissioned blockchain network, leveraging smart contracts to automate submission recording, reward allocation, and model-commerce operations while enhancing security, stability, and transparency.
- We provide rigorous convergence analysis under nonconvex, L-smooth assumptions and fairness guarantees, and validate our framework experimentally on benchmark datasets, demonstrating significant reductions in ownership disparity with accuracy competitive to FedAvg.

5.2 Design

5.2.1 Problem Formulation

In traditional FL, K clients each utilize their local datasets to collaboratively train a single global model θ_g. This process aims to solve the following global objective:

$$\min_{\theta_g} \; G\big(F_1(\theta_g), F_2(\theta_g), \ldots, F_K(\theta_g)\big), \tag{5.1}$$

where $F_i(\theta_g)$ denotes the local objective of client i, computed as the expected loss over its data distribution $\mathcal{D}_i$, i.e.,

$$F_i(\theta_g) = \mathbb{E}_{(x,y)\sim\mathcal{D}_i}[\mathcal{L}(\theta_g; x, y)]. \tag{5.2}$$

The function $G(\cdot)$ represents an aggregation rule for combining the local objectives. A typical choice is the simple average:

$$G(F_1, \ldots, F_K) = \frac{1}{K}\sum_{i=1}^{K} F_i(\theta_g). \tag{5.3}$$

In FedAvg, this objective is approached through iterative rounds of local training and global aggregation. At round r, each client i receives the global model θ_g^{r-1} and performs a local gradient update:

$$\theta_i^r = \theta_g^{r-1} - \eta\nabla F_i(\theta_g^{r-1}). \tag{5.4}$$

These locally updated models are then averaged to produce the new global model θ_g^r. This framework enables decentralized training while maintaining the privacy of client data.

Despite its practical benefits, FedAvg has notable fairness and security limitations. Clients with minimal data or computational effort contribute little to the global model, yet receive the same model as more substantial contributors. Moreover, adversarial clients may submit manipulated updates to degrade global performance. Without personalized aggregation, such vulnerabilities can compromise both fairness and robustness.

To address these issues, we propose a personalized aggregation mechanism where each client's influence on the global model is weighted in proportion to its training

utility. To quantify contribution in a privacy-preserving way, we require each client to submit a set of synthetically generated test records. These records are collectively used to construct an inclusive test set for evaluating all local models and determining their relative contributions, i.e, $\mathcal{T} = \bigsqcup_{i=1}^{K} \mathcal{T}_i$

5.2.2 Measuring Contribution

We measure the contribution of client i at round r based on the performance improvement of its local model θ_i^r relative to the current global model θ_g^r. The global improvement gap is given by:

$$\Delta_{\text{gap}} = \ell_g^{r-1} - \ell_i^r, \tag{5.5}$$

Where ℓ_g^{r-1} denotes the loss of the previous global model on a shared evaluation set $\mathcal{T}$, and ℓ_i^r is the loss of the client's local model on the same set. However, as training progresses, Δ_{gap} tends to diminish, leading to vanishing contribution scores.

To address this, we define a composite improvement score that also considers the local progress made by client i. Specifically, we compute:

$$\Delta_{\text{local}} = \ell_i^{r-1} - \ell_i^r, \tag{5.6}$$

$$\Delta_i = \alpha\, \Delta_{\text{gap}} + (1 - \alpha)\, \Delta_{\text{local}}, \tag{5.7}$$

Where $\alpha \in [0, 1]$ is a mixing coefficient that balances global and local contributions. If the combined score Δ_i is negative, indicating no meaningful improvement, the client's contribution is treated as negligible. We define the final contribution C_i as:

$$C_i = \max(\Delta_i, 0) + \epsilon, \tag{5.8}$$

Where $\epsilon > 0$ ensures a non-zero lower bound on the contribution, preserving continuity assumptions required for convergence guarantees.

5.2.3 Model Aggregation

We propose a novel aggregation mechanism that leverages the contribution scores C_i defined in the previous subsection. In this approach, the updated global model θ_g^{r+1} is computed as a weighted average of the local models θ_i^r, where the weights are determined by applying a softmax function over the contribution scores:

$$w_i = \frac{\exp(C_i)}{\sum_{j=1}^{K} \exp(C_j)}, \tag{5.9}$$

$$\theta_g^{r+1} = \sum_{i=1}^{K} w_i \, \theta_i^r. \tag{5.10}$$

By employing the softmax transformation, we ensure that clients contributing more meaningfully to the training process receive proportionally higher influence in the global model update. This adaptive weighting enhances both the fairness and robustness of the FL system, as it naturally emphasizes high-utility updates while diminishing the effect of less impactful or potentially adversarial ones.

5.2.4 Personalization Procedure

In our framework, personalization is designed to promote fairness by tailoring each client's model update in proportion to the significance of its contribution. Specifically,

clients with minimal contributions receive updates that preserve their prior local models, while those with substantial contributions are guided more closely toward the global model. This behavior is captured via a personalized interpolation:

$$\theta_i^{r+1} = (1 - \gamma_i)\, \theta_i^r + \gamma_i\, \theta_g^{r+1}, \tag{5.11}$$

$$\gamma_i = \min \left\{ \left(\frac{C_i}{\max_j C_j} \right)^p,\ \gamma_{\max} \right\}, \tag{5.12}$$

where $\gamma_i \in [0, 1]$ denotes the personalization coefficient for client i, derived from its normalized contribution C_i. The design of this update rule is motivated by three key considerations:

- **Relative Comparison:** The contribution of each client is normalized by the maximum contribution in the current round, ensuring that γ_i reflects each client's standing relative to others rather than in absolute terms.

- **Boosting Modest Contributions:** The exponent $p \in (0, 1)$ acts as a soft boost for smaller contributions. This allows clients with moderate improvements to still benefit meaningfully from the global model, especially in later rounds when absolute contribution scores tend to diminish.

- **Controlled Personalization:** The upper bound $\gamma_{\max}$ serves as a safeguard against excessive global influence by capping the interpolation factor. Allowing $\gamma_i = 1$ would overwrite the client's model with the global model entirely, breaking the continuity required for convergence analysis. By ensuring $\gamma_i < 1$, we preserve a minimal dependence on each client's previous state, enabling a smooth and analyzable contraction path toward the solution.

5.2.5 Model Acquisition

In our architecture, it is essential to account for scenarios where a client either lacks sufficient computational resources or possesses limited data, rendering its contribution to the training process minimal or infeasible. Additionally, certain clients may have participated extensively in earlier training rounds, such that their data distribution has already been well-integrated into the global model. As a result, further contributions from these clients offer diminishing returns.

We must also accommodate newly joined clients, who require access to the current global model before they can begin training. To address these challenges, we introduce a token-based mechanism through which clients can obtain the global model in exchange for a payment. The number of tokens paid determines the extent to which a client's local model is updated toward the global model, using a formula similar to the personalization mechanism described previously:

$$\theta_i^{r+1} = (1 - \beta(p))\,\theta_i^r + \beta(p)\,\theta_g^r, \tag{5.13}$$

$$\beta(p) = \max\left(\frac{p}{p^r_{\text{model}}},\ 1\right), \tag{5.14}$$

where p is the number of tokens paid by client i, and p^r_{model} denotes the current price of the global model at round r.

The global model price is initialized at a base value and incrementally updated based on the model's improvement in performance over time:

$$p^0_{\text{model}} = p^{\text{base}}_{\text{model}}, \tag{5.15}$$

$$p^r_{\text{model}} = p^{r-1}_{\text{model}} + \lambda\,(\ell^{r-1}_g - \ell^r_g), \tag{5.16}$$

Where ℓ^r_g is the global model loss at round r, and λ is a hyperparameter controlling the sensitivity of the model price to performance improvements.

5.2.6 Reward Distribution

In each training round r, a fixed total number of tokens R_{total} is distributed as a reward among the participating clients in proportion to their measured contribution. To mitigate sharp disparities and amplify smaller contributions, we apply a logarithmic transformation to the raw contribution values:

$$\tilde{C}_i = \log(1 + C_i), \tag{5.17}$$

$$r_i = \frac{\tilde{C}_i}{\sum_{j=1}^{K} \tilde{C}_j}, \tag{5.18}$$

$$R_i = R_{\text{total}} \cdot r_i, \tag{5.19}$$

where R_i denotes the reward allocated to client i, and $\tilde{C}_i$ is the smoothed contribution score.

These tokens can be used by clients in future rounds to purchase access to the global model—particularly in cases where they are unable to contribute effectively due to limited data or computational capacity. Alternatively, tokens may be traded with newly joined clients, thereby creating an incentive-aligned and self-sustaining FL economy.

5.2.7 Blockchain Incorporation

FL offers significant advantages as a distributed learning paradigm, including collaborative model training without direct data sharing, and scalable, parallelized computation across clients. However, FL also suffers from critical challenges related to security, stability, and fairness, which can hinder its practical deployment."

- **Security**: In an FL system, both clients and servers can act maliciously and undermine the system's security. Clients can communicate malicious model updates that, when aggregated, significantly degrade the performance of the global model. An FL server, on the other hand, can perform adversarial attacks such as Membership Inference Attacks (MIA) on the submitted client models. In FL, since the server operates as a black-box from the clients' perspective, it can act as it pleases to threaten the system's security.

- **Stability**: The FL workflow heavily relies on the centralized server's availability. Consequently, the server becomes a single point of failure, where in the case of its downfall, the system comes to a halt, and training progress and contributions may be lost..

- **Fairness**: In this chapter, we discuss fairness in the form of appropriate model ownership concerning clients' contributions. Nevertheless, another unfair possibility in traditional FL algorithms might arise since the server is free to favor some clients over others in the aggregation step.

The disadvantages discussed above arise from the centralized nature of FL frameworks. By replacing the server-client architecture with a blockchain network, we can

improve security, stability, and fairness–concerning fair model aggregation. In addition, our proposed personalization scheme is much aligned with the abilities of a blockchain network for the following reasons:

1. We propose a personalized FL framework where clients are rewarded in proportion to their training contributions. However, in traditional settings, model ownership remains centralized. Incorporating blockchain technology in this matter means removing this centralized ownership and enabling true decentralized ownership of the global model, proportionally distributed among participants based on their contribution.

2. We provide a scheme for model acquisition and reward distribution for training nodes. Both of such features are among the most notable use cases of blockchain technology. Hence, implementing our personalized FL algorithm seems best suited when combined with this decentralized network.

5.2.8 System Workflow

Integrating a blockchain network into our architecture decentralizes the server's responsibilities, distributing them across a suite of smart contracts. Despite this architectural shift, the overall workflow closely resembles that of traditional FL systems.

Our scheme is intended for a permissioned blockchain, where participants are authorized and known to each other. This choice is appropriate since our design is motivated by providing a solution for enterprise-level organizations with the possibility of training a superior model through decentralized and personalized FL. Hence, in

our design, we assume the number as well as the identity of the participating nodes are known to one another.

Under these assumptions, the workflow of our blockchain-enabled personalized federated learning (BPFL) design is demonstrated in Algorithm 5 and is as follows:

Algorithm 5 Blockchain-enabled Personalized Federated Learning (BPFL)

/ Executes on each client device */*

Ensure: ClientTrain():

for each round $r = 1, 2, \ldots, R$ **do**
  for each client i in parallel **do**
    if r == 1 **OR** just joined the network **then**
      Receive (or purchase) θ_g^r and set to θ_i^r
    end if
    for each batch b in epoch E **do**
      $\theta_i^r \leftarrow \theta_i^r - \eta \nabla F_i(\theta_i^r)$.
    end for
    Generate synthetic data $\mathcal{T}_i$ from $\mathcal{D}_i^t$
    Send $(\theta_i^r, \mathcal{T}_i)$ to the blockchain Ledger
    Receive personalized θ_i^{r+1} from the blockchain.
  end for
end for

Ensure: Aggregate($[\theta_i^r, \mathcal{T}_i$ for each client $i]$):

/* *Compute contributions* */

$\mathcal{T} = \bigsqcup_{i=1}^{K} \mathcal{T}_i$

for each client i **do**

$\Delta_{\text{gap}} = \ell_g^{r-1} - \ell_i^r$

$\Delta_{\text{local}} = \ell_i^{r-1} - \ell_i^r$

$\Delta_i = \alpha\Delta_{\text{gap}} + (1-\alpha)\Delta_{\text{local}}$

$C_i = \max(\Delta_i, 0) + \epsilon$

end for

$w_i = \frac{\exp(C_i)}{\sum_j \exp(C_j)}$

$\theta_g^r = \sum_i w_i \theta_i^r$

for each client i **do**

$\gamma_i = \min\left\{\left(\frac{C_i}{\max_j C_j}\right)^p, \gamma_{\max}\right\}$

$\theta_i^{r+1} = (1-\gamma_i)\,\theta_i^r + \gamma_i\,\theta_g^r$

end for

for each client i **do**

$\tilde{C}_i = \log(1 + C_i)$

$r_i = \frac{\tilde{C}_i}{\sum_j \tilde{C}_j}$

$R_i = R_{\text{total}} \cdot r_i$

end for

$p_{\text{model}}^r = p_{\text{model}}^{r-1} + \lambda\,(\ell_g^{r-1} - \ell_g^r)$

Return $[\theta_i^{r+1}, R_i, p_{\text{model}}^r]$

1. Each node receives the latest public initialization of the global model and performs local training using its private dataset.

2. Upon completion of local training, each node submits a reference (e.g., an address or IPFS hash) to its updated model to the blockchain. This submission is recorded along with the node's digital signature and wallet address.

3. The blockchain aggregates all submitted references and makes them available to a designated off-chain aggregator.

4. The off-chain aggregator downloads all submitted models, computes each client's contribution to global performance, aggregates the models using contribution-weighted methods, and personalizes the resulting model for each node based on their contribution.

5. The aggregator also computes monetary or token-based rewards for each participant and updates the market price of the resulting global model.

6. These computed values—personalized model references, reward distributions, and updated pricing—are then submitted to the blockchain. Smart contracts are triggered to update the balances of node wallets and store the current model state.

5.2.9 Off-chain Aggregator

A key design choice in this architecture is the introduction of a designated off-chain aggregator that maintains privileged access to the private global model and

performs computation-intensive tasks off-chain. While this component introduces an element of centralization, it is justified on the following grounds:

- Designated Worker with Verifiability: The off-chain aggregator acts as a semi-trusted computation unit, but its actions are made transparent and auditable through verifiable commitments recorded on-chain. For example, model hashes, contribution scores, and reward allocations can all be independently verified by participants. Moreover, each aggregation round's output is immutably recorded, ensuring accountability over time.

- Consortium-Based Operation: The aggregator can be implemented as a consortium service operated by multiple authorized participants, with aggregation outputs requiring multi-signature approval or quorum consensus. This mitigates the risk of unilateral control and provides a decentralized operational layer, aligned with the goals of fairness and transparency.

This hybrid architecture allows us to maintain the efficiency of off-chain computation while enforcing trust and accountability through the blockchain's immutable and transparent infrastructure. Furthermore, the private nature of the global model—visible only to the aggregator—prevents leakage of sensitive aggregate information, while personalized updates ensure that each client receives a model tuned to their data distribution and contribution. This design thus aligns the incentives of all participants while preserving security, stability, and fairness.

This workflow effectively decentralizes coordination, enforces fairness through programmable rules, and enhances the security and robustness of the learning process.

Furthermore, it provides a natural infrastructure for integrating incentive mechanisms, provenance tracking, and verifiable training contributions—key requirements for enterprise-grade FL systems.

5.3 Convergence Analysis

In this section, we present the convergence analysis of our proposed personalized FL algorithm. Traditionally, FL seeks to optimize a uniform global objective across K clients:

$$\min_{\theta_g} \; F_w(\theta_g) := \frac{1}{K} \sum_{i=1}^{K} F_i(\theta_g) \tag{5.20}$$

However, rather than optimizing this static average, we introduce a dynamic, contribution-weighted global objective that adapts over communication rounds. Specifically, in round r, the global model minimizes:

$$\min_{\theta_g} \; F_w^r(\theta_g) := \sum_{i=1}^{K} w_i^r F_i(\theta_g) \tag{5.21}$$

Unlike FedAvg, this global objective varies across rounds. As a result, we analyze convergence to a moving target, i.e., the time-varying objective $F_w^r(\theta_g)$. To proceed, we first establish a set of standard and architecture-specific assumptions and then derive convergence guarantees for both the global and personalized local models.

5.3.1 Assumptions

Assumptions 1-3 are standard in FL convergence analysis literature. However, Assumptions 4-6 are specific to our personalization architecture.

Assumption 1. *Each client i has a loss function $F_i(\theta) = \mathbb{E}_{(x,y)\sim\mathcal{D}_i}[\mathcal{L}(\theta; x, y)]$ which may be non-convex but is L-smooth:*

$$\|\nabla F_i(x) - \nabla F_i(y)\| \leq L\|x - y\|, \quad \forall x, y. \tag{5.22}$$

Remark 1. *We make no convexity assumption on the local objectives F_i. This reflects the non-convex nature of modern ML models, such as deep neural networks, which are commonly used in FL settings. Our analysis, therefore, applies to a broader class of practical problems where convexity does not hold.*

Assumption 2. *Since clients use stochastic gradients to update local models, we assume the variance of the stochastic gradient at each client is bounded:*

$$\mathbb{E}_{\xi\sim\mathcal{D}_i}\left[\|\nabla f_i(\theta; \xi) - \nabla f_i(\theta)\|^2\right] \leq \sigma^2 \tag{5.23}$$

Assumption 3. *We assume that the heterogeneity of local objectives is bounded in expectation, with respect to the contribution-based weights w_i. Specifically, for all x, we have:*

$$\sum_{i=1}^{K} w_i \|\nabla F_i(x) - \nabla F_w(x)\|^2 \leq \zeta^2, \tag{5.24}$$

where ζ quantifies the extent of data heterogeneity across clients.

Assumption 4. *We assume that both the clients' local losses and the global loss remain finite and bounded across all communication rounds:*

$$0 \leq \ell_i^r \leq L_{\max}, \quad \forall i, r \tag{5.25}$$

$$0 \leq \ell_g^r \leq L_{\max}, \quad \forall r \tag{5.26}$$

This assumption guarantees the following:

1. *The contribution scores C_i are upper bounded:*

$$C_i \leq \alpha L_{\max} + (1-\alpha)L_{\max} + \epsilon = L_{\max} + \epsilon \tag{5.27}$$

2. *Softmax weights w_i are both lower and upper bounded, improving numerical stability.*

Assumption 5. *Let the contribution score C_i^r be a function of the bounded and Lipschitz-continuous loss values $\ell_g^r, \ell_i^r, \ell_i^{r-1}$, and the aggregation weights w_i^r are defined via a softmax over contribution scores. Then, for all clients i and communication rounds r, there exists a constant $\beta > 0$ such that:*

$$\left| w_i^r - w_i^{r-1} \right| \leq \beta. \tag{5.28}$$

This implies that the aggregation weights vary smoothly with the temporal changes in the loss values, ensuring stable updates to the global model.

Assumption 6. *The personalization step for each client is defined as a convex combination between the local model and the global model:*

$$\theta_i^{r+1} = (1-\gamma_i)\theta_i^r + \gamma_i \theta_g^{r+1} \tag{5.29}$$

where $\gamma_i \in [0, \gamma_{\max}]$, and $0 < \gamma_{\max} < 1$. This ensures that local models move toward the global model in a contractive manner, promoting stability and bounding divergence across rounds.

5.3.2 Auxiliary Lemmas

Lemma 1 (Descent via Smoothness)**.** *Under Assumption 1 (L-smoothness), for any round r and any two iterates x, y,*

$$F_w^r(y) \;\le\; F_w^r(x) \;+\; \langle \nabla F_w^r(x),\, y - x \rangle \;+\; \frac{L}{2}\|y - x\|^2. \tag{5.30}$$

Proof. Direct from L-smoothness of each F_i and the convex combination defining F_w^r. □

Lemma 2 (Unbiasedness of the Global Update)**.** *Under the stochastic-gradient model and $\mathbb{E}[\xi_i^r] = 0$, the aggregation step*

$$\theta_g^{r+1} = \sum_{i=1}^{K} w_i^r \big(\theta_g^r - \eta(\nabla F_i(\theta_g^r) + \xi_i^r)\big) \tag{5.31}$$

satisfies

$$\mathbb{E}[\theta_g^{r+1} - \theta_g^r] = -\eta\, \nabla F_w^r(\theta_g^r). \tag{5.32}$$

Proof. Take expectation inside the sum and use $\sum_i w_i^r = 1$, $\mathbb{E}[\xi_i^r] = 0$. □

Lemma 3 (Second-Moment Bound)**.** *Under Assumption 2 (bounded variance σ^2), the step-length satisfies*

$$\mathbb{E}[\|\theta_g^{r+1} - \theta_g^r\|^2] \le \eta^2 \|\nabla F_w^r(\theta_g^r))\|^2 + \eta^2\sigma^2 \tag{5.33}$$

Proof. Expand $\theta_g^{r+1} - \theta_g^r = -\eta \sum_i w_i^r \nabla F_i(\theta_g^r) + \eta \sum_i w_i^r(-\xi_i^r)$, square, take expectation, drop the cross-term by independence, and bound the variance. □

Lemma 4 (Objective Drift)**.** *Under Assumption 5 (weight drift β) and bounded losses (Assumption 4),*

$$\big|F_w^{r+1}(\theta) - F_w^r(\theta)\big| \;\le\; K\beta\, L_{\max}, \quad \forall\theta. \tag{5.34}$$

Proof.

$$|F_w^{r+1}(\theta_g) - F_w^r(\theta_g)| = |\sum_{i=1}^{K}(w_i^{r+1} - w_i^r)F_i(\theta_g)|$$
$$\leq |\sum_{i=1}^{K}(w_i^{r+1} - w_i^r)| \cdot |F_i(\theta_g)| \leq K\beta L_{\max} \tag{5.35}$$

□

Lemma 5 (Tracking Error with Time-Varying γ_i^r)**.** *Under Assumption 6, for each client i and round $r \geq 1$, the tracking error satisfies:*

$$\|\theta_i^r - \theta_g^r\|^2 \leq (1 - \gamma_i^{r-1})^2\|\theta_i^{r-1} - \theta_g^{r-1}\|^2 + \|\theta_g^r - \theta_g^{r-1}\|^2. \tag{5.36}$$

Proof. By the personalization step and subtracting θ_g^r, we get:

$$\theta_i^r - \theta_g^r = (1 - \gamma_i^{r-1})(\theta_i^{r-1} - \theta_g^{r-1}) + (1 - \gamma_i^{r-1})(\theta_g^{r-1} - \theta_g^r).$$

Taking norms and applying standard inequality bounds gives the result. □

Lemma 6 (Lower Bound on Personalization Weight)**.** *Suppose each client's contribution score satisfies $C_i^r \geq \epsilon > 0$ for all r, and the personalization rate is defined as*

$$\gamma_i^r = \min\left\{\left(\frac{C_i^r}{\max_j C_j^r}\right)^p, \gamma_{\max}\right\}. \tag{5.37}$$

Then $\gamma_i^r \geq \bar{\gamma} > 0$, where

$$\bar{\gamma} := \min\left\{\left(\frac{\epsilon}{L_{\max} + \epsilon}\right)^p, \gamma_{\max}\right\}. \tag{5.38}$$

Proof. Since $C_i^r \geq \epsilon$ and $C_j^r \leq L_{\max} + \epsilon$ by Assumption 4, we get

$$\frac{C_i^r}{\max_j C_j^r} \geq \frac{\epsilon}{L_{\max} + \epsilon}, \tag{5.39}$$

Hence

$$\gamma_i^r \geq \left(\frac{\epsilon}{L_{\max} + \epsilon}\right)^p. \tag{5.40}$$

Clipping at $\gamma_{\max}$ still preserves this as a lower bound. □

5.3.3 Theorems

Theorem 1. *Let $\{\theta_g^r\}_{r=0}^R$ be the sequence of global models generated by Algorithm 5. Suppose Assumptions 1–6 hold, and choose a constant step size $\eta \leq 1/L$. Define the round-r surrogate objective*

$$F_w^r(\theta) \;:=\; \sum_{i=1}^{K} w_i^r \, F_i(\theta), \tag{5.41}$$

and let

$$F_w^* \;:=\; \inf_{0 \leq r \leq R} \inf_{\theta} \; F_w^r(\theta). \tag{5.42}$$

Then the average squared gradient norm of the time-varying objective satisfies

$$\frac{1}{R} \sum_{r=0}^{R-1} \mathbb{E}\big[\|\nabla F_w^r(\theta_g^r)\|^2\big] \;\leq\; \mathcal{O}\big(\frac{1}{R} + \sigma^2 + \beta\big). \tag{5.43}$$

Interpretation. *This suggests that as the algorithm progresses, the global model's parameter updates become smaller and the algorithm approaches a minimum or a stationary point.*

Proof. Start from Lemma 1 with $x = \theta_g^r$, $y = \theta_g^{r+1}$:

$$F_w^r(\theta_g^{r+1}) \leq F_w^r(\theta_g^r) + \langle \nabla F_w^r(\theta_g^r), \theta_g^{r+1} - \theta_g^r \rangle + \frac{L}{2}\|\theta_g^{r+1} - \theta_g^r\|^2 \tag{5.44}$$

Take total expectation, substitute Lemmas 2 and 3:

$$\mathbb{E}[F_w^r(\theta_g^{r+1})] \leq F_w^r(\theta_g^r) + -\eta\|\nabla F_w^r(\theta_g^r)\|^2 + \frac{L}{2}(\eta^2\|\nabla F_i(\theta_g^r))\|^2 + \eta^2\sigma^2)$$

$$\mathbb{E}[F_w^r(\theta_g^{r+1})] \leq F_w^r(\theta_g^r) - \eta(1 - \frac{L\eta}{2})\|\nabla F_w^r(\theta_g^r)\|^2 + \frac{L}{2}(\eta^2\sigma^2)$$

Next, account for the change in objectives across rounds via Lemma 4:

$$\mathbb{E}[F_w^{r+1}(\theta_g^{r+1})] \leq \mathbb{E}[F_w^r(\theta_g^{r+1})] + K\beta L_{\max}. \tag{5.45}$$

Combining the two gives, for each r:

$$\mathbb{E}[F_w^r(\theta_g^{r+1})] \leq F_w^r(\theta_g^r) - \eta(1 - \frac{L\eta}{2})\|\nabla F_w^r(\theta_g^r)\|^2 + \frac{L}{2}(\eta^2\sigma^2) + K\beta L_{\max}$$

Summing from $r = 0$ to $R - 1$ and telescoping:

$$\mathbb{E}[F_w(\theta_g^R)] - F_w(\theta_g^0) \leq -\eta\left(1 - \frac{L\eta}{2}\right)\sum_{r=0}^{R-1}\|\nabla F_w(\theta_g^r)\|^2 + R\left(\frac{L}{2}(\eta^2\sigma^2) + K\beta L_{\max}\right)$$

Rearrange:

$$\frac{1}{R}\sum_{r=0}^{R-1}\mathbb{E}\left[\|\nabla F_w(\theta_g^r)\|^2\right] \leq \frac{F_w(\theta_g^0) - \mathbb{E}[F_w(\theta_g^R)]}{\eta\left(1 - \frac{L\eta}{2}\right)R} + \frac{L}{2}(\eta^2\sigma^2) + K\beta L_{\max}$$

And, finally:

$$\frac{1}{R}\sum_{r=0}^{R-1}\mathbb{E}\big[\|\nabla F_w^r(\theta_g^r)\|^2\big] \;\leq\; \mathcal{O}\big(\frac{1}{R} + \sigma^2 + \beta\big). \tag{5.46}$$

□

Theorem 2. *Let the personalization step follow:*

$$\theta_i^{r+1} = (1 - \gamma_i^r)\theta_i^r + \gamma_i^r\theta_g^{r+1} \tag{5.47}$$

where γ_i^r varies over time and satisfies $\gamma_i^r \in [\bar{\gamma}, \gamma_{\max}]$ for some $\bar{\gamma} > 0$, as shown in Lemma 7.

Assume the global model drift is bounded as:

$$\mathbb{E}[\|\theta_g^{r+1} - \theta_g^r\|^2] \leq \zeta_g^2. \tag{5.48}$$

Then the expected personalized tracking error satisfies:

$$\mathbb{E}[\|\theta_i^r - \theta_g^r\|^2] \leq (1 - \bar{\gamma})^{2r}\|\theta_i^0 - \theta_g^0\|^2 + \frac{\zeta_g^2}{\bar{\gamma}}. \tag{5.49}$$

Interpretation. *This theorem provides a bound on the personalized tracking error of a client model in FL. It shows that as the personalization coefficient γ_i^r is adjusted over time, the difference between the personalized model and the global model decreases exponentially, ensuring that personalized models also converge to the global model or a neighborhood depending on the personalization level.*

Proof. From the personalization update:

$$\theta_i^{r+1} = (1-\gamma_i^r)\theta_i^r + \gamma_i^r \theta_g^{r+1}, \tag{5.50}$$

subtracting θ_g^{r+1}, we have:

$$\theta_i^{r+1} - \theta_g^{r+1} = (1-\gamma_i^r)(\theta_i^r - \theta_g^r) + (1-\gamma_i^r)(\theta_g^r - \theta_g^{r+1}). \tag{5.51}$$

Applying Lemma 5, we obtain:

$$\|\theta_i^{r+1} - \theta_g^{r+1}\|^2 \leq (1-\gamma_i^r)^2 \|\theta_i^r - \theta_g^r\|^2 + \|\theta_g^{r+1} - \theta_g^r\|^2. \tag{5.52}$$

Taking expectations and using bounded drift $\mathbb{E}[\|\theta_g^{r+1} - \theta_g^r\|^2] \leq \zeta_g^2$:

$$\mathbb{E}[\|\theta_i^{r+1} - \theta_g^{r+1}\|^2] \leq (1-\gamma_i^r)^2 \mathbb{E}[\|\theta_i^r - \theta_g^r\|^2] + \zeta_g^2. \tag{5.53}$$

Unrolling the recurrence:

$$\mathbb{E}[\|\theta_i^r - \theta_g^r\|^2] \leq \left(\prod_{k=0}^{r-1}(1-\gamma_i^k)^2\right) \|\theta_i^0 - \theta_g^0\|^2 + \zeta_g^2 \sum_{t=1}^{r} \prod_{k=t}^{r-1}(1-\gamma_i^k)^2.$$

Since $\gamma_i^r \geq \bar{\gamma} > 0$ (Lemma 6), we have:

$$\prod_{k=0}^{r-1}(1-\gamma_i^k)^2 \leq (1-\bar{\gamma})^{2r}, \quad \sum_{t=1}^{r} \prod_{k=t}^{r-1}(1-\gamma_i^k)^2 \leq \frac{1}{\bar{\gamma}}. \tag{5.54}$$

Hence:

$$\mathbb{E}[\|\theta_i^r - \theta_g^r\|^2] \leq (1-\bar{\gamma})^{2r} \|\theta_i^0 - \theta_g^0\|^2 + \frac{\zeta_g^2}{\bar{\gamma}}. \tag{5.55}$$

□

5.4 Discussions

5.4.1 Personalization Trade-offs

Personalization Cap

A central component of BPFL is the personalization of each local model based on its contribution value. As shown in Equation 5.12, this interpolation is modulated by the coefficient γ_i, which is in turn upper-bounded by the cap value $\gamma_{\max}$. In this section, we elaborate on the significance of $\gamma_{\max}$ and provide guidance on how it can be effectively tuned.

The cap $\gamma_{\max}$ plays a crucial role in the personalization mechanism for two key reasons:

1. **Convergence Assurance:** The cap is essential for satisfying Assumption 6 and Lemma 5, which form the foundation of Theorem 2—our convergence guarantee for local model trajectories. To uphold this theoretical guarantee while still allowing substantial global influence, we recommend setting $\gamma_{\max} \in [0.9, 0.99]$.

2. **Personalization–Globalization Trade-off:** The value of $\gamma_{\max}$ can be viewed as a tunable knob that controls the balance between personalization and globalization. In nearly IID settings, a higher $\gamma_{\max}$ encourages closer alignment with the global model, which can accelerate convergence and improve generalization. Conversely, in highly heterogeneous environments, a lower $\gamma_{\max}$ helps preserve the individuality of local models, mitigating the risk of performance degradation due to over-generalization.

The value of $\gamma_{\max}$ should be selected based on both the theoretical requirements of convergence and the characteristics of the underlying data distribution. We propose the following practical guidelines:

- **Convergence Guarantees:** Theoretical analysis (see Theorem 2) relies on each client retaining a minimal divergence from the global model. To uphold this condition, $\gamma_{\max}$ should be strictly less than 1. Empirically, values in the range $[0.9, 0.99]$ provide a good balance between convergence speed and stability.

- **IID settings:** When client datasets are homogeneous, higher values of $\gamma_{\max}$ (e.g., 0.95 or above) are preferable. These promote stronger alignment with the global model, accelerating convergence and leveraging the benefits of shared structure.

- **Non-IID settings:** When client data distributions vary significantly, a lower $\gamma_{\max}$ (e.g., 0.7–0.9) may be more appropriate. This allows clients to retain more of their unique local characteristics, improving local performance and reducing the negative effects of global overfitting.

- **Empirical tuning:** Just as any other hyperparameter $\gamma_{\max}$ is best selected through a fine-tuning process. In practice, we recommend starting with a default value such as $\gamma_{\max} = 0.95$, and adjusting based on convergence diagnostics and validation accuracy across clients. Grid search or adaptive tuning schemes may also be used to optimize performance.

Personalization Exponent

Another key variable in our personalization scheme is the personalization exponent p. This exponent serves to amplify client contributions as the model converges, particularly when composite contributions from clients start to diminish. To maintain this effect over time, dynamically adjusting p is a natural choice. This can be achieved by initially setting p at the higher end of the $[0, 1]$ range and applying a custom decay function—such as exponential or linear decay—to gradually reduce its value.

5.4.2 Convergence Speed

The convergence speed of FedAvg on non-convex and non-IID data is $\mathcal{O}\left(\frac{1}{R}\right)$ [81], which is inherently faster than that of BPFL, whose convergence rate is $\mathcal{O}\left(\frac{1}{R} + \sigma^2 + \beta\right)$. However, this gap narrows in scenarios characterized by:

- **Increased Data Heterogeneity**: As the data distributions across clients become more heterogeneous (i.e., clients have more diverse, non-IID data), the benefits of BPFL become more pronounced. The personalization parameter γ_i allows for more controlled updates, preventing large, destabilizing changes from clients whose data is highly skewed or unrepresentative. This helps in reducing the variance σ^2 and the contribution term β, leading to more stable updates and faster convergence to personalized optima.

- **Dissimilar Client Capabilities**: In settings where clients vary significantly in computational power, dataset size, or learning rates, BPFL's personalized

update mechanism (via adaptive γ_i) ensures that clients with limited resources or less relevant data exert less influence on the global model. This prevents the model from drifting toward suboptimal regions and reduces oscillations, thereby improving convergence in cases where FedAvg may struggle due to its equal-weight aggregation strategy.

5.4.3 Malicious Nodes

The BPFL design is intended for creating a collaborative learning environment with fair model ownership among a group of training nodes. Due to the authentication of all participants, we assume that there are no adversaries deliberately trying to diminish global model performance by communicating malicious model updates or other forms of attacks. Nevertheless, our design inherently offers safeguards against ineffective and unproductive model updates by evaluating each model update against a robust, synthetically generated, and diverse test dataset. Consequently, only the most effective local models contribute to the global update, ensuring both robustness and efficiency in the learning process.

5.5 Experimental Setups

In this section, we introduce the settings and environments we selected to evaluate BPFT. The section is divided into two parts. In the first subsection, we discuss problem-specific conditions such as data allocations for proper validation of BPFT; in the second subsection, we introduce the tools, conditions, and hyperparameters we

chose during our evaluation[1].

5.5.1 Data Allocation Schemes

This chapter introduces an algorithm that fairly incentivizes training clients in a blockchain-enabled federated learning system by granting model ownership proportional to their training contributions. To evaluate our framework, we simulate clients with varying contribution capacities and record the resulting outcomes. In our experiments, we associate training contribution with dataset size, under the assumption that a larger dataset corresponds to a greater contribution. Accordingly, we implement three distinct data allocation schemes: uniform, linear, and quadratic.

1. Uniform Allocation: Each node receives an equal share of the total dataset. Let the total dataset size be N and the number of participating nodes be K. Then, each node i (for $i = 0, 1, \ldots, K-1$) receives:

$$D_i^{\text{uniform}} = \frac{N}{K}, \quad \text{or equivalently,} \quad \frac{D_i^{\text{uniform}}}{N} = \frac{1}{K}. \tag{5.56}$$

2. Linear Allocation: In a linear scheme, the data allocated to each node increases linearly with its index. That is, node i receives a fraction proportional to $(i+1)$:

$$\frac{D_i^{\text{linear}}}{N} = \frac{i+1}{\sum_{j=0}^{K-1}(j+1)} = \frac{i+1}{\frac{K(K+1)}{2}}. \tag{5.57}$$

For instance, with $K = 4$, the respective fractions are $\frac{1}{10}$, $\frac{2}{10}$, $\frac{3}{10}$, $\frac{4}{10}$.

3. Quadratic Allocation: In the quadratic scheme, each node i receives data

[1]The following link provides the implementation code for our design: **https://github.com/amirrezaskh/BPFL**

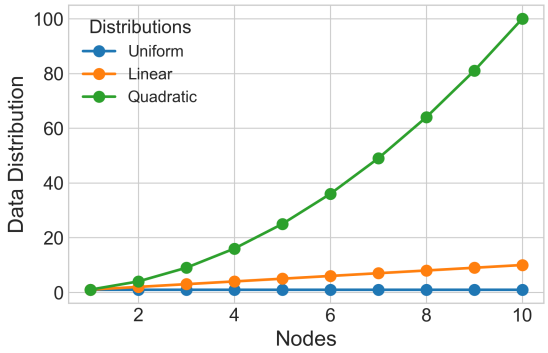


Figure 5.1: BPFL Data Allocation Schemes.

in proportion to $(i+1)^2$:

$$\frac{D_i^{\text{quadratic}}}{N} = \frac{(i+1)^2}{\sum_{j=0}^{K-1}(j+1)^2} = \frac{(i+1)^2}{\frac{K(K+1)(2K+1)}{6}}. \tag{5.58}$$

For $K = 4$, this yields: $\frac{1^2}{30}$, $\frac{2^2}{30}$, $\frac{3^2}{30}$, $\frac{4^2}{30}$.

As shown in Figure 5.1, the gap between allocations widens with increasing index values. These distinct schemes allow us to model clients with varying dataset sizes, and by extension, differing training contributions.

5.5.2 General Settings

To benchmark our framework, we compare it against the traditional FedAvg algorithm across all experiments. Both algorithms are evaluated under the three data allocations using four widely adopted datasets: MNIST, FashionMNIST, CIFAR10, and CIFAR100. For MNIST and FMNIST, we employ LeNet5; for CIFAR10 and CIFAR100, we use ResNet18.

All experiments are conducted with four participating nodes. This limited scale is intentional, reflecting our focus on collaborative training among a small set of

Table 5.1: BPFL Hyperparameters

Name	**Symbol**	**Value**
Number of Clients	K	4
Local Learning Rate	η	10^{-3}
Weight Decay	ν	10^{-4}
Contribution Mixing Coefficient	α	0.5
Contribution Lower Bound	ϵ	10^{-9}
Personalization Exponent	p	0.5
Max Personalization Factor	$\gamma_{\max}$	0.95, 0.7
Total Reward Tokens	R_{total}	300
Base Model Price	$p_{\text{model}}^{\text{base}}$	50
Price Sensitivity Coefficient	λ	10
Number of Rounds	R	20
Local Epochs	E	5
Batch Size	B	32

known organizations. Accordingly, scalability is not considered in this evaluation. All local datasets are kept non-IID to reflect realistic heterogeneity among clients. Hyperparameters are held constant across all runs to ensure consistent comparisons.

The proposed system is implemented using the Hyperledger Fabric framework to construct the blockchain and smart contracts. As an enterprise permissioned blockchain with incredible modularity, this platform fits perfectly into our design. Each training node is built by integrating PyTorch (for model training) and Flask (as the communication interface with the blockchain network). Nodes operate as separate processes in our experimental environment, which is hosted on a machine equipped with an Intel Xeon(R) 4216 CPU (16 cores, 32 threads) and an NVIDIA GeForce RTX 3080 GPU for accelerated training.

We summarize all hyperparameters used in our FL framework in Table 5.1. For

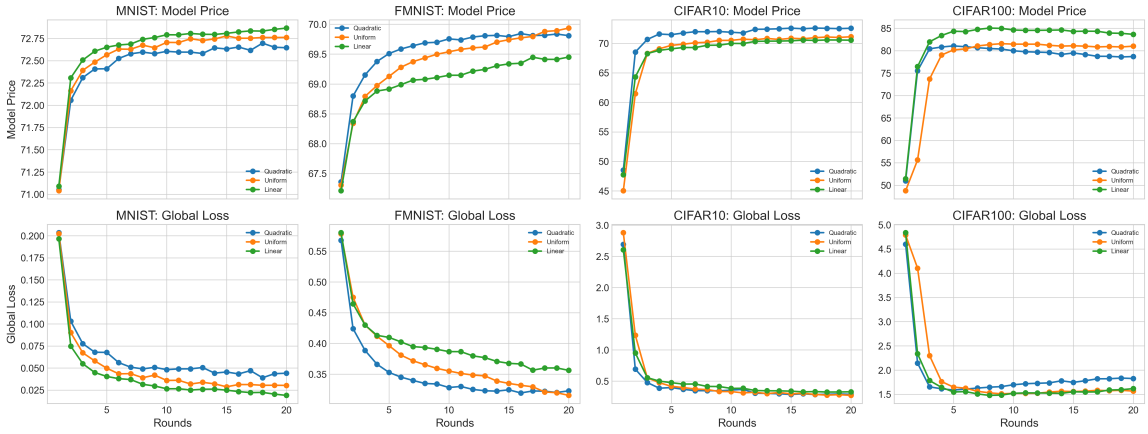


Figure 5.2: BPFL Global Loss and Model Price for LeNet5 and ResNet18 across different data allocations on MNIST, FMNIST, CIFAR10, and CIFAR100.

local model training, we employ a learning rate of $\eta = 10^{-3}$ with a weight decay of $\nu = 10^{-4}$. Each client trains locally for $E = 5$ epochs using mini-batches of size $B = 32$. The total number of clients is $K = 4$, and the global training proceeds for $R = 20$ rounds.

Our aggregation mechanism balances global and local contributions using a mixing coefficient $\alpha = 0.5$. To avoid vanishing influence, we enforce a minimum contribution of $\epsilon = 10^{-9}$. The personalization factor is once capped at $\gamma_{\max} = 0.95$ and once at $\gamma_{\max} = 0.7$, and the personalization exponent is $p = 0.5$. In our incentive mechanism, we distribute $R_{\text{total}} = 300$ reward tokens, with a base model price $p_{\text{model}}^{\text{base}} = 50$ and a price sensitivity coefficient $\lambda = 10$.

Table 5.2: Comparison of BPFL and FedAvg results across different datasets and data allocations

Dataset	**Data Allocation**					
	Uniform		**Linear**		**Quadratic**	
	BPFL	**FedAvg**	**BPFL**	**FedAvg**	**BPFL**	**FedAvg**
MNIST	0.0287	0.0323	0.0189	0.0486	0.0389	0.0371
FMNIST	0.3154	0.3480	0.3561	0.3506	0.3193	0.3718
CIFAR10	0.2707	0.3216	0.3245	0.3263	0.2846	0.2912
CIFAR100	1.5107	1.4762	1.4781	1.4667	1.5890	1.5412

5.6 Results

This section provides the results of running BPFL under a multitude of conditions, such a varying datasets and data allocations to evaluate global and local model performance, fair model ownership, and rewards.

5.6.1 BPFL vs FedAvg

Since we introduce a novel model aggregation scheme, we must present a performance comparison between our solution and the baseline FedAvg. We have performed this task by running all our experiments both with BPFL and FedAvg. To provide an appropriate comparison between the two algorithms, we have selected $\gamma_{\max} = 0.95$ since it greatly affects the global model's performance. In addition, we provide a comparison of the convergence speed of these two algorithms.

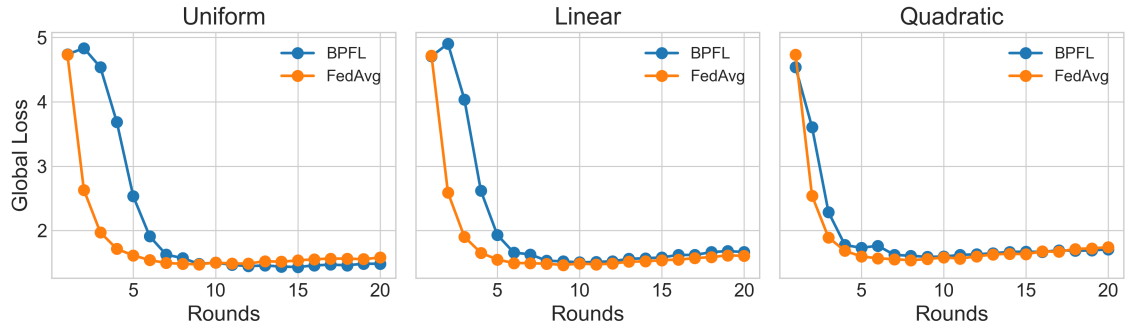


Figure 5.3: BPFL and FedAvg Convergence Speed Comparison on ResNet18 trained with CIFAR100 For All Data Allocations

Performance

Table 5.2 compares the global loss values of BPFL and FedAvg across four datasets (MNIST, FMNIST, CIFAR10, and CIFAR100) under three data allocation schemes, Uniform, Linear, and Quadratic using for four nodes. In each variation, two values are reported, with the better one highlighted in light green. According to these results, BPFL outperforms FedAvg, achieving an average loss improvement of 8.5% across all experiments. These results are encouraging, especially considering that the primary goal of BPFL was not to surpass FedAvg in performance but to introduce a fair model ownership mechanism.

Convergence Speed

As discussed in Section 5.4, FedAvg is inherently designed to converge faster than BPFL; however, the gap in convergence speed narrows as the variation in client contribution capabilities increases.

Table 5.3: BPFL Personalized Model Results Across Different Datasets and Data Allocations

Dataset	**Data Allocation**							
	Linear				**Quadratic**			
	N1	**N2**	**N3**	**N4**	**N1**	**N2**	**N3**	**N4**
MNIST	0.0872	0.0704	0.0739	0.0705	0.1288	0.0699	0.0536	0.0550
FMNIST	0.3401	0.3240	0.3194	0.3121	0.4927	0.3642	0.3571	0.3514
CIFAR10	0.5026	0.4936	0.4432	0.4240	0.6803	0.5291	0.4113	0.3391
CIFAR100	2.2591	2.0711	2.0140	1.9954	2.3172	2.1298	2.0318	1.8601

Table 5.4: BPFL Reward Distribution Between Nodes in Different Datasets and Data Allocations

Dataset	**Data Allocation**							
	Linear				**Quadratic**			
	N1	**N2**	**N3**	**N4**	**N1**	**N2**	**N3**	**N4**
MNIST	388.5	2067.5	2354.0	1189.8	361.3	209.7	2355.2	3073.6
FMNIST	868.3	1017.8	1327.1	2786.6	230.8	964.6	2423.6	2380.8
CIFAR10	906.6	1092.1	1351.3	2649.8	974.7	843.4	1326.6	2855.1
CIFAR100	1373.9	1542.8	1503.1	1580.0	1375.4	1392.1	1444.3	1788.0

As demonstrated in Figure 5.3, as the difference between client data sizes becomes greater, the convergence speed of BPFL enhances and gets closer to FedAvg.

5.6.2 Personalized Model Performances

BPFL personalizes each client's model in proportion to its contribution, which is computed based on a combination of local and global loss improvements. As data partition sizes—and consequently, training contributions—become more varied, the performance of client models is expected to improve. In addition to benefiting from

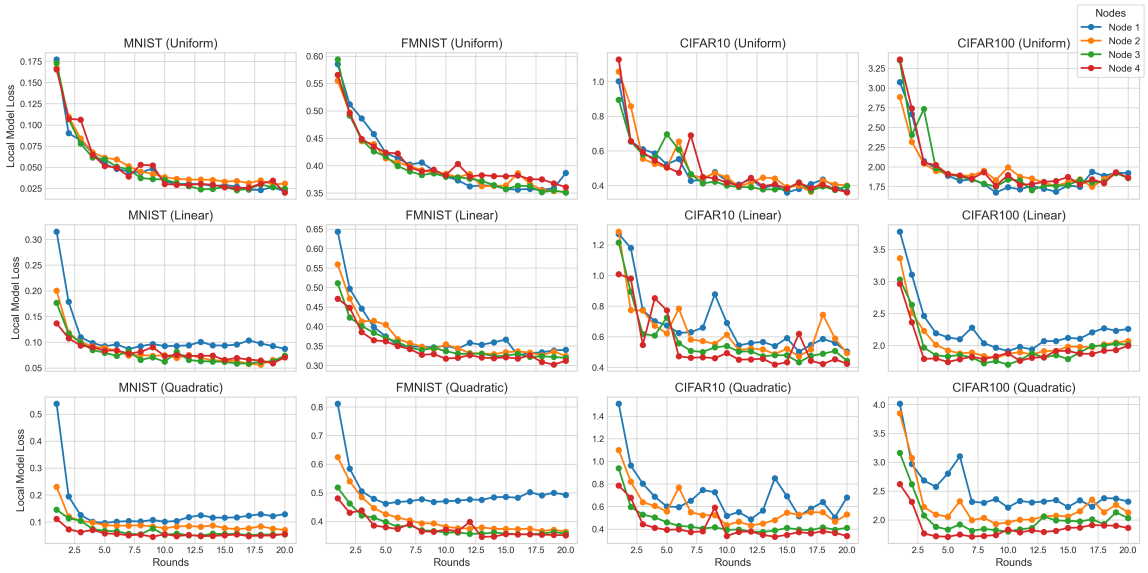


Figure 5.4: Comparison of Local Model Losses Across Datasets Under Varying Data Allocations

more accurate models, clients with larger data partitions also receive greater rewards over the course of training.

The following results are based on experiments conducted with a personalization cap of $\gamma_{\max} = 0.7$. This choice increases the degree of personalization applied to local models, thereby amplifying their individual effects.

Table 5.3 reports personalized loss values for all nodes across the four datasets under two data allocation schemes–Linear and Quadratic. As evidenced in the table, nodes with larger datasets and higher indexes achieve better model performance. This reflects the fairness in personalization intended by our design.

Table 5.4 shows the final rewards received by all nodes under the same conditions. Nodes with greater data volumes receive higher rewards, as their models contribute more significantly to the global model and are therefore more highly incentivized.

5.6.3 Global Model Convergence

In this section, we provide the convergence of local and global models throughout all training rounds. This demonstration

Figure 5.2 illustrates global loss and model price trends across 20 training rounds for all datasets and data allocations. As seen in the figure, both model price and global loss improve over training rounds, indicating convergence to a global minimum. In addition, in the case of more complex datasets such as CIFAR10 and CIFAR100, the performance of all data allocation schemes is considerably close to one another, suggesting that our aggregation scheme converges regardless of varying training contributions.

Figure 5.4 displays local loss trends over time. In linear and quadratic data allocations, increased data disparities–nodes with higher indexes-lead to greater divergence in local model performance, validating the effectiveness of the proposed personalization scheme.

5.7 Conclusion

In this chapter, we introduced a novel BPFL framework that empowers industry-level organizations to participate in a fair and collaborative learning process. BPFL addresses the fairness limitations of traditional FedAvg by introducing a contribution metric and personalizing each local model based on this value. The framework also employs blockchain technology to replace the centralized server, mitigating associated risks related to security and fairness. This design aligns with our token-based

mechanism, which allows participants with insufficient contributions to acquire the global model through token exchange. Our theoretical analysis under nonconvex, L-smooth assumptions establishes both convergence and fairness guarantees, and our experiments on benchmark datasets demonstrate significant reductions in ownership disparity while maintaining accuracy comparable to FedAvg.

Chapter 6

Conclusion

In this thesis, we explored three key dimensions of integrating blockchain technology with collaborative learning algorithms. First, we introduced a novel consensus mechanism—Proof-of-Collaborative-Learning (PoCL)—which repurposes the computational power traditionally wasted in blockchain mining toward FL tasks, thereby aligning security with meaningful computation. Next, we proposed Sharded SplitFed Learning (SSFL), an enhancement to the standard SFL framework that distributes the communication and computation burden of the central server across multiple parallel shards. Building upon SSFL, we developed Blockchain-enabled SplitFed Learning (BSFL)—the first fully decentralized SFL architecture—where a committee-based consensus mechanism governs training, validation, and aggregation through smart contracts. Finally, we addressed the fairness issue in FL by introducing Blockchain-enabled Personalized Federated Learning (BPFL), which incorporates a novel contribution metric and an aggregation mechanism that proportionally rewards clients based on their utility. To promote equitable access, we further extended BPFL with

a token-based model acquisition system, allowing clients to obtain the global model in exchange for tokens. We conclude with a theoretical convergence analysis of BPFL under standard non-convex optimization assumptions, providing formal guarantees for its effectiveness.

6.1 Future Works

In the following, we discuss approaches to advance scalability, privacy, and the evaluation mechanism of the three proposed algorithms:

- **Employing Differential Privacy:** In all three designs, we ensure a private FL or SFL training procedure; nevertheless, privacy guarantees can be further strengthened by integrating differential privacy. In this way, we prevent any concerns about the aggregator or other components in the system from breaching privacy using MIA attacks. Differential privacy can be achieved by modifying the standard gradient descent step or adding noise to each local model before submission to the ledger.

- **Alternative Evaluation Metric:** In all three algorithms, an evaluation mechanism is utilized, either in the form of validation in the committee consensus or computing contribution for personalization. These evaluation schemes are built based on the loss value of the submitted models; nevertheless, the evaluation mechanism can be adapted to metrics beyond traditional losses, especially for generative applications where labels are not required for assessing model performance. In these scenarios, metrics such as Feature Likelihood Divergence (FLD)

[82] or Fréchet Inception Distance (FID) [83] can be adopted. These metrics are particularly relevant for evaluating nodes in tasks involving generative models, offering better insights into their performance.

- **SL Variations:** In BSFL and SSFL, the model is split into two parts. A promising extension involves partitioning the model into three or more segments, enabling clients to train the final layers locally. This would eliminate the need to share batch targets with the SL server. Both SSFL and BSFL can be extended to support multi-part model splits. Such an extension could improve client-side privacy and reduce data exposure while maintaining the framework's scalability and performance. Nevertheless, this setup increases the computational power demanded from the clients since they must contribute to the training of more layers. This trade-off encourages further assessment from the distributed learning research community.

- **Layer-wise Personalization:** The current BPFL design tailors the entire global model based on client contribution. However, this can be computationally expensive for larger, denser models (e.g., LLMs). A more efficient alternative is layer-wise personalization, where deeper personalization (i.e., more layers updated) is applied for clients with higher contributions. This approach balances fairness and computational efficiency.

Bibliography

[1] J. Gantz and D. Reinsel, "The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east," *IDC iView: IDC Analyze the future*, vol. 2007, no. 2012, pp. 1–16, 2012.

[2] C. Meurisch and M. Mühlhäuser, "Data protection in ai services: A survey," *ACM Computing Surveys (CSUR)*, vol. 54, no. 2, pp. 1–38, 2021.

[3] L. Sweeney, "k-anonymity: a model for protecting privacy," *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, vol. 10, no. 5, p. 557–570, Oct. 2002. [Online]. Available: https://doi.org/10.1142/S0218488502001648

[4] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, ser. STOC '09. New York, NY, USA: Association for Computing Machinery, 2009, p. 169–178. [Online]. Available: https://doi.org/10.1145/1536414.1536440

[5] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[6] O. Gupta and R. Raskar, "Distributed learning of deep neural network over multiple agents," *Journal of Network and Computer Applications*, vol. 116, pp. 1–8, 2018.

[7] P. Vepakomma, O. Gupta, T. Swedish, and R. Raskar, "Split learning for health: Distributed deep learning without sharing raw patient data," *arXiv preprint arXiv:1812.00564*, 2018.

[8] C. Thapa, P. C. M. Arachchige, S. Camtepe, and L. Sun, "Splitfed: When federated learning meets split learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 8, 2022, pp. 8485–8493.

[9] A. Sokhankhosh and S. Rouhani, "Proof-of-collaborative-learning: A multi-winner federated learning consensus algorithm," in *2024 IEEE International Conference on Blockchain (Blockchain)*, 2024, pp. 370–377.

[10] S. Nakamoto. (2017) Bitcoin: A peer-to-peer electronic cash system. [Online]. Available: http://www.bitcoin.org/bitcoin.pdf

[11] Y. Qu, M. P. Uddin, C. Gan, Y. Xiang, L. Gao, and J. Yearwood, "Blockchain-enabled federated learning: A survey," *ACM Computing Surveys*, vol. 55, no. 4, pp. 1–35, 2022.

[12] C. Natoli, J. Yu, V. Gramoli, and P. Esteves-Verissimo, "Deconstructing blockchains: A comprehensive survey on consensus, membership and structure," *arXiv preprint arXiv:1908.08316*, 2019.

[13] J. Zhu, J. Cao, D. Saxena, S. Jiang, and H. Ferradi, "Blockchain-empowered federated learning: Challenges, solutions, and future directions," *ACM Comput. Surv.*, vol. 55, no. 11, feb 2023. [Online]. Available: https://doi.org/10.1145/3570953

[14] S. Jiang, J. Cao, H. Wu, Y. Yang, M. Ma, and J. He, "Blochie: a blockchain-based platform for healthcare information exchange," in *2018 ieee international conference on smart computing (smartcomp)*. IEEE, 2018, pp. 49–56.

[15] C. T. Nguyen, D. T. Hoang, D. N. Nguyen, D. Niyato, H. T. Nguyen, and E. Dutkiewicz, "Proof-of-stake consensus mechanisms for future blockchain networks: Fundamentals, applications and opportunities," *IEEE Access*, vol. 7, pp. 85 727–85 745, 2019.

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

[17] B. Lashkari and P. Musilek, "A comprehensive review of blockchain consensus mechanisms," *IEEE Access*, vol. 9, pp. 43 620–43 652, 2021.

[18] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," in *2014 USENIX annual technical conference (USENIX ATC 14)*, 2014, pp. 305–319.

[19] M. Castro, B. Liskov *et al.*, "Practical byzantine fault tolerance," in *OsDI*, vol. 99, no. 1999, 1999, pp. 173–186.

[20] X. Xu, I. Weber, and M. Staples, "Architecture for blockchain applications," 2019.

[21] V. Buterin *et al.*, "Ethereum white paper," *GitHub repository*, vol. 1, no. 22-23, pp. 5–7, 2013.

[22] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich *et al.*, "Hyperledger fabric: a distributed operating system for permissioned blockchains," in *Proceedings of the thirteenth EuroSys conference*, 2018, pp. 1–15.

[23] M. Hao, H. Li, G. Xu, S. Liu, and H. Yang, "Towards efficient and privacy-preserving federated deep learning," in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, 2019, pp. 1–6.

[24] B. Li, Y. Wu, J. Song, R. Lu, T. Li, and L. Zhao, "Deepfed: Federated deep learning for intrusion detection in industrial cyber–physical systems," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 8, pp. 5615–5624, 2021.

[25] B. Yin, H. Yin, Y. Wu, and Z. Jiang, "Fdc: A secure federated deep learning mechanism for data collaborations in the internet of things," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6348–6359, 2020.

[26] M. A. Ferrag, O. Friha, L. Maglaras, H. Janicke, and L. Shu, "Federated deep learning for cyber security in the internet of things: Concepts, applications, and experimental analysis," *IEEE Access*, vol. 9, pp. 138 509–138 542, 2021.

[27] Q. Duan, S. Hu, R. Deng, and Z. Lu, "Combined federated and split learning in

edge computing for ubiquitous intelligence in internet of things: State-of-the-art and future directions," *Sensors*, vol. 22, no. 16, p. 5983, 2022.

[28] Y. Gao, M. Kim, S. Abuadbba, Y. Kim, C. Thapa, K. Kim, S. A. Camtep, H. Kim, and S. Nepal, "End-to-end evaluation of federated learning and split learning for internet of things," in *2020 International Symposium on Reliable Distributed Systems (SRDS)*, 2020, pp. 91–100.

[29] I. Ceballos, V. Sharma, E. Mugica, A. Singh, A. Roman, P. Vepakomma, and R. Raskar, "Splitnn-driven vertical partitioning," *arXiv preprint arXiv:2008.04137*, 2020.

[30] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *2017 IEEE symposium on security and privacy (SP)*. IEEE, 2017, pp. 3–18.

[31] Y. Li, C. Chen, N. Liu, H. Huang, Z. Zheng, and Q. Yan, "A blockchain-based decentralized federated learning framework with committee consensus," *IEEE Network*, vol. 35, no. 1, pp. 234–241, 2021.

[32] S. Truex, N. Baracaldo, A. Anwar, T. Steinke, H. Ludwig, R. Zhang, and Y. Zhou, "A hybrid approach to privacy-preserving federated learning," in *Proceedings of the 12th ACM workshop on artificial intelligence and security*, 2019, pp. 1–11.

[33] X. Qu, S. Wang, Q. Hu, and X. Cheng, "Proof of federated learning: A novel

energy-recycling consensus algorithm," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 8, pp. 2074–2085, 2021.

[34] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, "Blockchain and federated learning for privacy-preserved data sharing in industrial iot," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 4177–4186, 2020.

[35] S. Yuan, B. Cao, M. Peng, and Y. Sun, "Chainsfl: Blockchain-driven federated learning from design to realization," in *2021 IEEE Wireless Communications and Networking Conference (WCNC)*, 2021, pp. 1–6.

[36] R. N. Alief, M. A. Paramartha Putra, A. Gohil, J.-M. Lee, and D.-S. Kim, "Flb2: Layer 2 blockchain implementation scheme on federated learning technique," in *2023 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, 2023, pp. 846–850.

[37] Y. Zhang, Y. Tang, Z. Zhang, M. Li, Z. Li, S. Khan, H. Chen, and G. Cheng, "Blockchain-based practical and privacy-preserving federated learning with verifiable fairness," *Mathematics*, vol. 11, no. 5, p. 1091, 2023.

[38] Z. Wang, N. Dong, J. Sun, W. Knottenbelt, and Y. Guo, "zkfl: Zero-knowledge proof-based gradient aggregation for federated learning," *IEEE Transactions on Big Data*, pp. 1–14, 2024.

[39] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, "Blockchain and federated learning for privacy-preserved data sharing in industrial iot," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 4177–4186, 2020.

[40] W. Zhang, Q. Lu, Q. Yu, Z. Li, Y. Liu, S. K. Lo, S. Chen, X. Xu, and L. Zhu, "Blockchain-based federated learning for device failure detection in industrial iot," *IEEE Internet of Things Journal*, vol. 8, no. 7, pp. 5926–5937, 2021.

[41] Y. Qu, S. R. Pokhrel, S. Garg, L. Gao, and Y. Xiang, "A blockchained federated learning framework for cognitive computing in industry 4.0 networks," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 4, pp. 2964–2973, 2021.

[42] R. Kumar, A. A. Khan, J. Kumar, Zakria, N. A. Golilarz, S. Zhang, Y. Ting, C. Zheng, and W. Wang, "Blockchain-federated-learning and deep learning models for covid-19 detection using ct imaging," *IEEE Sensors Journal*, vol. 21, no. 14, pp. 16 301–16 314, 2021.

[43] M. A. Rahman, M. S. Hossain, M. S. Islam, N. A. Alrajeh, and G. Muhammad, "Secure and provenance enhanced internet of health things framework: A blockchain managed federated learning approach," *IEEE Access*, vol. 8, pp. 205 071–205 087, 2020.

[44] L. Cui, X. Su, Z. Ming, Z. Chen, S. Yang, Y. Zhou, and W. Xiao, "Creat: Blockchain-assisted compression algorithm of federated learning for content caching in edge computing," *IEEE Internet of Things Journal*, vol. 9, no. 16, pp. 14 151–14 161, 2022.

[45] S. R. Pokhrel and J. Choi, "Federated learning with blockchain for autonomous vehicles: Analysis and design challenges," *IEEE Transactions on Communications*, vol. 68, no. 8, pp. 4734–4746, 2020.

[46] A. O. Bada, A. Damianou, C. M. Angelopoulos, and V. Katos, "Towards a green blockchain: A review of consensus mechanisms and their energy consumption," in *2021 17th international conference on distributed computing in sensor systems (DCOSS)*. IEEE, 2021, pp. 503–511.

[47] (2024) Clamcoin. [Online]. Available: https://clamcoin.org/#

[48] L. Ren, "Proof of stake velocity: Building the social currency of the digital age," 2014. [Online]. Available: https://api.semanticscholar.org/CorpusID:14946615

[49] F. Yang, W. Zhou, Q. Wu, R. Long, N. N. Xiong, and M. Zhou, "Delegated proof of stake with downgrade: A secure and efficient blockchain consensus algorithm with downgrade mechanism," *IEEE Access*, vol. 7, pp. 118 541–118 555, 2019.

[50] (2024) Shield. [Online]. Available: https://whitepaper.io/document/296/shield-1-whitepaper

[51] S. King and S. Nadal, "Ppcoin: Peer-to-peer crypto-currency with proof-of-stake," 2012. [Online]. Available: https://api.semanticscholar.org/CorpusID:42319203

[52] W. Li, S. Andreina, J.-M. Bohli, and G. Karame, "Securing proof-of-stake blockchain protocols," 09 2017, pp. 297–315.

[53] (2024) Vericonomy. [Online]. Available: https://vericonomy.com/

[54] K. Karantias, A. Kiayias, and D. Zindros, "Proof-of-burn," in *Financial Cryptography and Data Security: 24th International Conference, FC 2020, Kota Kin-*

abalu, Malaysia, February 10–14, 2020 Revised Selected Papers 24. Springer, 2020, pp. 523–540.

[55] G. Ateniese, I. Bonacina, A. Faonio, and N. Galesi, "Proofs of space: When space is of the essence," in *Proc. Int. Conf. Secur. Cryptogr. Netw.* Amalfi, Italy: Springer, 2014, pp. 538–557.

[56] A. Yakovenko, "Solana: A new architecture for a high performance blockchain," Solana Labs, Tech. Rep., 2018.

[57] I. Bentov, C. Lee, A. Mizrahi, and M. Rosenfeld, "Proof of activity: Extending bitcoin's proof of work via proof of stake [extended abstract]y," *SIGMETRICS Perform. Eval. Rev.*, vol. 42, no. 3, p. 34–37, dec 2014. [Online]. Available: https://doi.org/10.1145/2695533.2695545

[58] I. Bentov, R. Pass, and E. Shi, "Snow white: Provably secure proofs of stake," *IACR Cryptol. ePrint Arch.*, vol. 2016, p. 919, 2016. [Online]. Available: https://api.semanticscholar.org/CorpusID:16970691

[59] X. Qu, S. Wang, Q. Hu, and X. Cheng, "Proof of federated learning: A novel energy-recycling consensus algorithm," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 8, pp. 2074–2085, 2021.

[60] A. Shoker, "Sustainable blockchain through proof of exercise," in *2017 IEEE 16th International Symposium on Network Computing and Applications (NCA)*, 2017, pp. 1–9.

[61] Z. Dong, Y. C. Lee, and A. Y. Zomaya. (2019) Proofware: Proof of useful

work blockchain consensus protocol for decentralized applications. [Online]. Available: http://arxiv.org/abs/1903.09276

[62] S. King, "Primecoin: Cryptocurrency with prime number proof-of-work," *Cryptology ePrint Archive*, vol. 1, no. 6, 2013.

[63] F. Bravo-Marquez, S. Reeves, and M. Ugarte, "Proof-of-learning: A blockchain consensus mechanism based on machine learning competitions," *2019 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPCON)*, pp. 119–124, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:199440961

[64] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, "Blockchain and federated learning for privacy-preserved data sharing in industrial iot," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 4177–4186, 2020.

[65] S. Oh, J. Park, P. Vepakomma, S. Baek, R. Raskar, M. Bennis, and S.-L. Kim, "Locfedmix-sl: Localize, federate, and mix for improved scalability, convergence, and latency in split learning," in *Proceedings of the ACM Web Conference 2022*, 2022, pp. 3347–3357.

[66] S. Pal, M. Uniyal, J. Park, P. Vepakomma, R. Raskar, M. Bennis, M. Jeon, and J. Choi, "Server-side local gradient averaging and learning rate acceleration for scalable split learning," *arXiv preprint arXiv:2112.05929*, 2021.

[67] S. Oh, H. Nam, J. Park, P. Vepakomma, R. Raskar, M. Bennis, and S.-L. Kim,

"Mix2sfl: Two-way mixup for scalable, accurate, and communication-efficient split federated learning," *IEEE Transactions on Big Data*, 2023.

[68] Z. Batool, K. Zhang, Z. Zhu, S. Aravamuthan, and U. Aivodji, "Block-fest: A blockchain-based federated anomaly detection framework with computation offloading using transformers," in *2022 IEEE 1st Global Emerging Technology Blockchain Forum: Blockchain Beyond (iGETblockchain)*, 2022, pp. 1–6.

[69] S. Sai and V. Chamola, "A blockchain-enabled split learning framework with a novel client selection method for collaborative learning in smart healthcare," *IEEE Transactions on Consumer Electronics*, pp. 1–1, 2024.

[70] L. Lyu, X. Xu, Q. Wang, and H. Yu, "Collaborative fairness in federated learning," *Federated Learning: Privacy and Incentive*, pp. 189–204, 2020.

[71] X. Xu and L. Lyu, "A reputation mechanism is all you need: Collaborative fairness and adversarial robustness in federated learning," *arXiv preprint arXiv:2011.10464*, 2020.

[72] W. Gao, G. Xu, and X. Meng, "Fedcon: Scalable and efficient federated learning via contribution-based aggregation," *Electronics*, vol. 14, no. 5, p. 1024, 2025.

[73] (2024) digiconomist. [Online]. Available: https://digiconomist.net/bitcoin-energy-consumption

[74] M. Shen, H. Wang, B. Zhang, L. Zhu, K. Xu, Q. Li, and X. Du, "Exploiting unintended property leakage in blockchain-assisted federated learning for intelligent

edge computing," *IEEE Internet of Things Journal*, vol. 8, no. 4, pp. 2265–2275, 2021.

[75] M. Radovanović, A. Nanopoulos, and M. Ivanović, "Nearest neighbors in high-dimensional data: The emergence and influence of hubs," in *Proceedings of the 26th Annual International Conference on Machine Learning*, 2009, pp. 865–872.

[76] N. Aloysius and M. Geetha, "A review on deep convolutional neural networks," in *2017 international conference on communication and signal processing (ICCSP)*. IEEE, 2017, pp. 0588–0592.

[77] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, and Y. Gao, "A survey on federated learning," *Knowledge-Based Systems*, vol. 216, p. 106775, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950705121000381

[78] D. Pasquini, G. Ateniese, and M. Bernaschi, "Unleashing the tiger: Inference attacks on split learning," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021, pp. 2113–2129.

[79] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, "Advances and open problems in federated learning," *Foundations and trends® in machine learning*, vol. 14, no. 1–2, pp. 1–210, 2021.

[80] Y. Fraboni, R. Vidal, and M. Lorenzi, "Free-rider attacks on model aggregation

in federated learning," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2021, pp. 1846–1854.

[81] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," *arXiv preprint arXiv:1907.02189*, 2019.

[82] M. Jiralerspong, J. Bose, I. Gemp, C. Qin, Y. Bachrach, and G. Gidel, "Feature likelihood score: Evaluating the generalization of generative models using samples," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[83] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," *Advances in neural information processing systems*, vol. 30, 2017.