Decentralized and Incentivized Federated Learning: A Blockchain-Enabled Framework Utilising Compressed Soft-Labels and Peer Consistency

Leon Witt, Usama Zafar, KuoYeh Shen, Felix Sattler, Dan Li, Songtao Wang, Wojciech Samek*, *Member, IEEE*,

Abstract—Federated Learning (FL) has emerged as a powerful paradigm in Artificial Intelligence, facilitating the parallel training of Artificial Neural Networks on edge devices while safeguarding data privacy. Nonetheless, to encourage widespread adoption, Federated Learning Frameworks (FLFs) must tackle (i) the power imbalance between a central authority and its participants, and (ii) the challenge of equitably measuring and incentivizing contributions. Existing approaches to decentralize and incentivize FL processes are hindered by (i) computational overhead and (ii) uncertainty in contribution assessment [1], limiting FL's scalability beyond use cases where trust between participants and the server is established. This work introduces a cutting-edge, blockchain-enabled federated learning framework that incorporates Federated Knowledge Distillation (FD) with compressed 1-bit soft-labels, aggregated through a smart contract. Furthermore, we present the Peer Truth Serum for Federated Distillation (PTSFD), which cultivates an incentive-compatible ecosystem by rewarding honest participation based on an implicit yet effective comparison of worker contributions. The primary innovation stems from its lightweight architecture that simultaneously promotes decentralization and incentivization, addressing critical challenges in contemporary FL approaches.

Index Terms—Federated Learning, Blockchain, Reward Mechanism, Federated Distillation, Decentralized Machine Learning.

1 Introduction

The ascent of *Machine Learning* (ML) has been marked by a growing emphasis on decentralized and privacy-preserving solutions. One of the leading solutions, *Federated Learning* (FL), allows training of *Deep Neural Networks* (NNs) across distributed devices, ensuring data remains localized, hence addressing privacy concerns. *Federated Averaging* (`FedAvg`) [2], a cornerstone algorithm in FL, achieves this by aggregating locally trained models to produce a global model. However, FL's transformative potential is curtailed by (i) A trust deficit emanating from the imbalance of power between workers and the central server and (ii) an absent practical reward mechanism to incentivize worker contributions. Notably, while blockchain's inherent transparency and immutability characteristics hold promise in addressing the trust issue, its effective integration with FL for scalable deployments has remained elusive [3], [4], [5]. Furthermore, designing mechanisms that effectively reward worker contributions without compromising data privacy remains open research in FL [1], [6]. Comparing and evaluating worker contributions in `FedAvg` is non-trivial since data always stays private [1], [7]. Traditional solutions like the Leave-one-out [8] or Shapley value [9], [10] introduce computational overhead and hinge on a centralized authority, constraining their adoption in decentralized, blockchain-based solutions. Lastly, the prohibitive cost of storing vast amounts of data on blockchain systems, compounded by the intensive computational demands, means popular methods like `FedAvg` struggle to fit within *General Purpose Blockchain Systems* (GPBS). This has prompted researchers towards *Application Specific Blockchain Systems* (ASBS) or off-chain aggregation [11], [12], [13], [14], [15] – both bringing their set of challenges.

- *L. Witt is with Tsinghua University, Beijing, China, and with the Department of Artificial Intelligence, Fraunhofer Heinrich Hertz Institute, 10587 Berlin, Germany. E-mail: leonmaximilianwitt@gmail.com*
- *U. Zafar is with Department of Information Technology, Uppsala University, Uppsala, Sweden.*
- *K. Shen, and D. Li are with Tsinghua University, Beijing, China.*
- *F. Sattler is with the Department of Artificial Intelligence, Fraunhofer Heinrich Hertz Institute, 10587 Berlin, Germany.*
- *S. Wang is with Zhongguancun Laboratory, Beijing, China.*
- *W. Samek is with the Department of Electrical Engineering and Computer Science, Technical University of Berlin, 10587 Berlin, Germany, with the Department of Artificial Intelligence, Fraunhofer Heinrich Hertz Institute, 10587 Berlin, Germany, and with BIFOLD – Berlin Institute for the Foundations of Learning and Data, 10587 Berlin, Germany. E-mail: wojciech.samek@hhi.fraunhofer.de*

This work was supported by the German Federal Ministry of Education and Research (BMBF) through the BIFOLD - Berlin Institute for the Foundations of Learning and Data (ref. 01IS18025A and ref. 01IS18037I) and the EU's Horizon 2020 project COPA EUROPE (no. 957059).

1.1 Contributions

In response to these challenges, this work introduces the Peer Truth Serum for Federated Distillation (PTSFD), a blockchain-enabled and incentivized FL framework.

It utilizes *Federated Knowledge Distillation* (FD) on 1-bit compressed soft-labels, combined with the Peer Truth Serum for Crowdsourcing [16], which is adjusted for the FD case.

1) **Incentivization:** We present the *Peer Truth Serum for Federated Distillation* (PTSFD), an informed-truthful multi-task peer prediction mechanism tailored for the FD case. It discerns contributions based on the correlation of reported 1-bit compressed soft-labels. This approach drastically reduces storage requirements, making it particularly suited for blockchain.
2) **Decentralization:** The reduced storage requirement and the simplicity of our method promote decentralization. The framework can be deployed on simple smart contracts hosted on prevalent blockchains, such as the Ethereum Virtual Machine [3], eliminating the need for specialized ASBS and simplifying the entire FL process.
3) **Scalability and Efficiency:** Our method, which builds upon Federated Knowledge Distillation, significantly reduces both communication overheads and blockchain storage needs. This sets the stage for large-scale, practical FL deployments without sacrificing efficiency or scalability.

We substantiate our contributions through theoretical validations and exhaustive experimental analyses. Our findings reveal a system that maintains a strong incentive-compatible equilibrium, demonstrating resilience against adversarial actions. Moreover, it showcases efficiency gains in storage and communication costs compared to `FedAvg` in various FL scenarios. The core of this work lies in its pioneering architecture, laying the groundwork for a lightweight, fully decentralized, incentivized, and efficient Federated Learning paradigm.

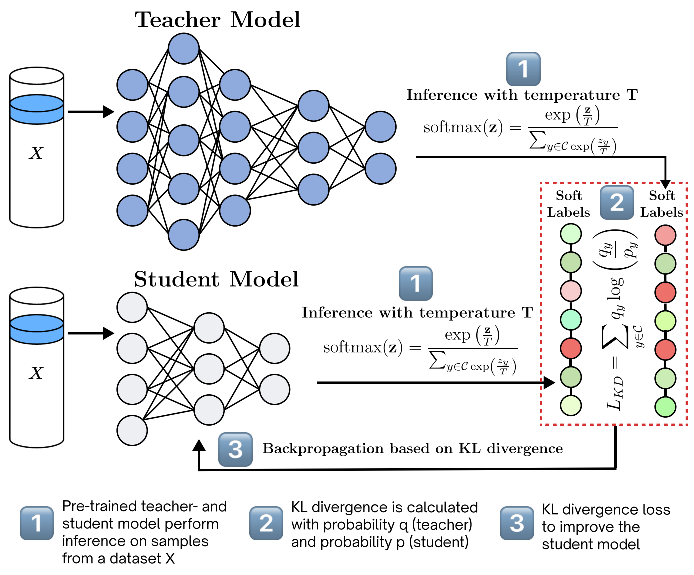
Teacher Model
Student Model
X
1 Inference with temperature T
$\mathrm{softmax}(\mathbf{z}) = \frac{\exp\left(\frac{\mathbf{z}}{T}\right)}{\sum_{y\in\mathcal{C}}\exp\left(\frac{z_y}{T}\right)}$
Soft Labels
2
$L_{KD} = \sum_{y\in\mathcal{C}} q_y \log\left(\frac{q_y}{p_y}\right)$
3 Backpropagation based on KL divergence
1 Pre-trained teacher- and student model perform inference on samples from a dataset X
2 KL divergence is calculated with probability q (teacher) and probability p (student)
3 KL divergence loss to improve the student model

Fig. 1: Knowledge Distillation

2 Background and Related Work

2.1 Federated Averaging

The most common algorithmic approach to FL problems is `FedAvg`, where the training process consists of an iteration of the following steps:

1) The central server selects a subset of clients $\mathcal{W}$, which participate in this training round.
2) The central server sends the current model θ to the selected clients.
3) The selected clients perform local training on their private data, leading to updated client models θ_i.
4) The updated models $\theta_i \; \forall i \in \mathcal{W}'$ are send back to the central server.
5) The central server aggregates the updated models to a new global model.

This training paradigm requires a two-way communication of the model θ (resp. θ_i) at every iteration, which can result in significant communication overhead for state-of-the-art NN models with hundreds of millions of parameters. To address this challenge, various approaches have been proposed, including pruning methods [17] and advanced compression techniques [18], [19], [20], [21], [22], [23], [24], [25]. However, despite these advances, the fundamental issue of scaling `FedAvg` to larger models remains, impeding the utilization of blockchain for storing or aggregating models [1].

2.2 Knowledge Distillation and Federated Settings

2.2.1 Knowledge Distillation

Knowledge Distillation (KD), depicted in Figure 1, is a technique in deep learning where a smaller NN model (often called the ”student”) is trained to mimic the behavior of a larger, pre-trained model (referred to as the ”teacher”) [27]. This is accomplished not by transferring the model parameters directly, but rather by aligning the output distributions of both models. Traditional training methods involve training a model directly on ground-truth labels, using a cross-entropy loss that measures the discrepancy between the model's predictions and these true labels. In contrast, KD employs a divergence-based loss, such as the Kullback-Leibler (KL) divergence, to measure the difference between the student's predicted probabilities and those of the teacher model. This divergence provides insights into how closely the student is able to mimic the behavior of its teacher. A distinct feature of KD is the use of ”softened” labels. In traditional classification tasks, hard labels are used, which unequivocally classify a data point into one category. However, the teacher model in KD provides ”soft” labels in the form of probabilities, indicating the confidence levels across various categories. These probabilities can be further softened using a temperature parameter T to yield a smoother distribution, capturing the nuances of decision boundaries and offering richer guidance to the student model. This process allows the student to inherit not just the overt knowledge from the ground-truth labels but also the implicit, or ”dark”, knowledge embedded in the teacher model's predictions. Since only soft-labels are necessary to perform backpropagation, models with

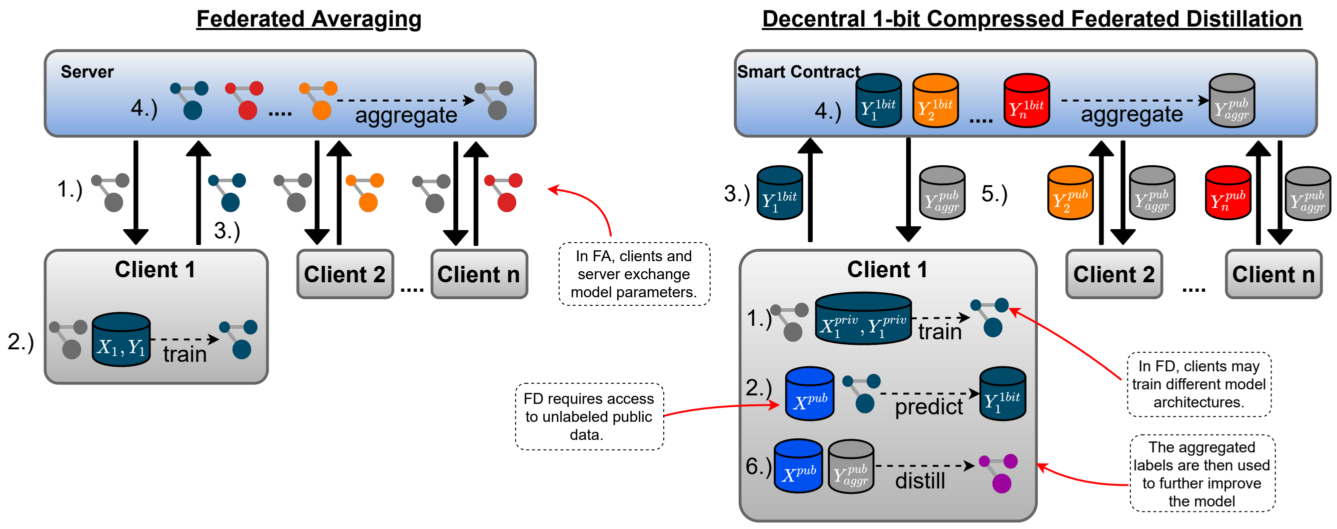
Federated Averaging
Server
4.)
aggregate
1.)
3.)
2.)
Client 1
X_1, Y_1
train
Client 2
Client n
In FA, clients and server exchange model parameters.
Decentral 1-bit Compressed Federated Distillation
Smart Contract
4.)
Y_1^{1bit}
Y_2^{1bit}
Y_n^{1bit}
aggregate
Y_{aggr}^{pub}
3.)
5.)
Y_2^{pub}
Y_n^{pub}
Client 1
1.)
X_1^{priv}, Y_1^{priv}
train
2.)
X^{pub}
predict
6.)
distill
FD requires access to unlabeled public data.
In FD, clients may train different model architectures.
The aggregated labels are then used to further improve the model

Fig. 2: Federated Learning (from [26]) vs. Decentral 1-bit Compressed Federated Distillation on Blockchain.

varying architectures can learn from the teacher. The appeal of KD lies in its ability to produce compact models with performance that closely mirrors that of much larger networks. These compact models are advantageous for deployment in resource-constrained environments, such as mobile devices or edge devices, without sacrificing much in terms of accuracy.With the above foundation in KD, we can delve deeper into its application in the FL setup, specifically focusing on Federated Distillation.

2.2.2 *Federated Distillation*

Drawing from the KD paradigm, Federated Distillation (FD) [28], [29], [30] extends the soft-label philosophy to a federated landscape, where the aggregated soft-logits from workers act akin to the soft-label output from an overarching teacher model. While `FedAvg` directly communicates the model parameters to transfer information between the central server and the clients, FD uses soft-label predictions Y_i^{pub} obtained on a separate public distillation dataset X^{pub} for this purpose. More precisely, the locally updated model $f_{\theta_i+\Delta\theta_i}$ is indirectly communicated to the central server by sending its predictions on the distillation dataset, i.e.,

$$Y_i^{pub} = \{f_{\theta_i+\Delta\theta_i}(x) \mid x \in X^{pub}\}. \quad (1)$$

Therefore, unlike traditional FL techniques such as FedAvg that mandate a consistent model structure across clients due to the aggregation of model parameter updates, FD does not require a single NN architecture but allows each worker to adopt a distinct architecture that might be best suited to its local data- or computational restrictions. Additionally, instead of with NN parameters, it scales with the size of the distillation dataset. This characteristic of FD can lead to communication savings [26], especially for large models. In this work, we modify a recently proposed, highly communication-efficient FD method [26], called *Compressed Federated Distillation (CFD)*, which is based on the multi-round protocol developed in [28], [30]. In our modified version of CFD, every client performs the following steps in each communication round:

1) **Train** on local datasets and improve model $\theta_i = \theta + \Delta\theta_i$ using X_i^{priv}, Y_i^{priv}.
2) **Predict labels** Y_i^{1bit} using the improved model θ_i on X^{pub} to compute soft-labels Y_i^{pub} and perform 1-bit quantization $Y_i^{1bit} = Q^{1bit}(Y_i^{pub})$.
3) **Upload** the integer-encoded compressed soft-labels to the smart contract (in a two-step commit-reveal fashion outlined in Algorithm 3).
4) **(Blockchain) Aggregate** predictions Y_{aggr}^{pub} by majority vote over all Y_i^{pub}.
5) **Download** the aggregated predictions Y_{aggr}^{pub} from the blockchain.
6) **Distill** the current model θ using X^{pub} and Y_{aggr}^{pub}.

The authors of [26] showed that CFD largely reduces the information necessary for exchange by quantization Q and the use of a small public distillation dataset (e.g., random subset selection). The savings are in the order of two orders of magnitude when compared to Federated Distillation, and more than four orders of magnitude when compared to `FedAvg`. The possibility to apply binary soft-label quantization, i.e., Q_b with $b = 1$, ensure three important properties for a decentralized CFD on Blockchain, namely

- It reduces the amount of information processed in the aggregation process heavily.
- It makes contributions by workers explicit and comparable.
- It supersedes the need for additional encryption like noise inducing Differential Privacy or computational heavy secure multiparty computation.

2.3 Blockchain Technology in FL context

Blockchain was initially introduced with Bitcoin by Satoshi Nakamoto in 2008 [31]. It is referred to as a distributed ledger managed by nodes in a peer-to-peer network, where cryptographic links of information ensure resistance to modification and immutability. The network is governed by a consensus mechanism [32] among peers, which supersedes the need for central coordination. The advent of general-purpose blockchains [3], with smart contract functionality supporting Turing-completeness, allows for a decentralized, immutable, and transparent business logic atop of blockchain. This technology is able to mitigate open problems of FL environments due to its inherent properties, namely:

Decentralization. In server-worker architectures, workers are exposed to a power imbalance and a single point of failure. A malicious server could (i) exclude workers arbitrarily or (ii) withhold reward payments. Furthermore, a server-worker design is not suitable for an environment where multiple entities share a common and equal interest in advancing their respective models. The decentral property of blockchain systems ensures a federal system for entities with equal power without the need for a central server.

Transparency and Immutability. Since every peer in the system shares the same data, data on blockchain can only be updated and never deleted. A transparent and immutable reward logic in an FL context ensures trust on the worker side. On the other hand, each worker is audited and can therefore be held accountable for malicious behavior.

Cryptocurrency. Many general-purpose blockchain systems come with cryptocurrency functionality, e.g., the option to implement payment schemes within the business logic of the smart contract. Based on a reward mechanism of the FL system, workers can be rewarded immediately, automatically, and deterministically without the need for a trusted third party.

To analyze Blockchain systems, we categorize them into two main types:

1) **Application Specific Blockchain Systems (ASBS):** Blockchains which have to be adapted to a specific FL use-case require a novel infrastructure. This causes overhead in terms of complexity at the development, deployment, and hence more likely to introduce vulnerabilities.
2) **General Purpose Blockchain Systems (GPBS):** These are limited due to restricted virtual machines and predefined consensus layers, but allow for easy development, deployment, and operation utilizing already existing frameworks [4], [5], [33], [34], [35].

These types can either be public or permissioned/private. Public blockchains, like Ethereum, are open networks where anyone can participate, hence making it expansive to use as every transaction has to be duplicated by every node in the network. Permissioned blockchains, such as Hyperledger Fabric [5], restrict participation to authorized entities, offering a controlled, efficient, and private environment that may be preferable for FL scenarios with known and trusted participants.

2.4 Related Work

We focus on related Federated Learning Frameworks (FLF) that (i) are both decentral *and* reward participation as well as (ii) use blockchain at its core to decentralize FL. That is, parameters are aggregated *or* stored in a decentralized way.[1] We extended the systematic analysis established by [1] to compare FL, the application of Blockchain and the Contribution measurement in Table 1. Note that the inherent complexity of FLF leads to heterogeneity in terms of application, overall design, special focus and details. [36] designs an FL system for home appliances using blockchain and a new normalization technique for differential privacy. Similarly, [42] introduces a regional FL framework for vehicles, integrating a reputation mechanism and a blockchain-secured trading platform. Focusing on robust mechanism deisigns, [15], [43] propose an FL protocol on blockchain using contest theory for worker engagement. [44] employs a Stackelberg game-based FL system considering contributions, deadlines, and upload times. [45] introduces DeepChain, a blockchain-secured FL framework with a special focus on privacy. [46] presents a two-layered blockchain for mobile edge networks. [47] minimizes communication costs in IoT FL through a double-layer aggregation model. [48] offers a specialized Democratic Learning (DemL) solution for on-device learning, including a unique consensus mechanism. [49] introduces Proof of FL (PoFL), an energy-efficient blockchain mechanism. [50] provides a secure FL framework for UAV-assisted sensing, incorporating differential privacy and reinforcement learning-based incentives. [51] designs a Mobile Crowdsensing framework that uses blockchain and edge intelligence for resource-constrained environments. [52] evaluates participant contributions transparently in its FL framework. [41] combines FL and blockchain for secure data sharing in neural training, using Shapley values for fair rewards. Lastly, [53] integrates blockchain and model distillation to accommodate model heterogeneity and enhance communication efficiency, yet it falls short in detailing blockchain operations and providing a theoretical analysis.

2.4.1 *Incentivization of FLF*

Measuring contributions in FL to fairly reward clients remains an open research challenge [1], [6]. Various metrics and methods are currently used for this purpose, each with its own set of challenges and limitations. [44], [48], [50], rely on self-reported information such as data size to determine rewards. However, this approach is susceptible to malicious behavior as false reporting leads to a maximal return. Alternatives include using similarity measures like the Euclidean distance of model updates [36], or employing voting systems for contribution assessment [43], [47]. Despite their utility, these methods lack rigorous theoretical and experimental validation and are vulnerable to attacks. Explicit methods like Shapley value [41], [52] or simple test-set accuracy [42], [49] have been utilized for explicit reward measurement. However, when applied in a decentralized context, these explicit methods (i) either require complex adjustments to the blockchain consensus mechanism, (ii) cause infeasible overhead (especially Shapley value), or require a central authority that measures the contribution against the test set, introducing a single point of failure.

1. often, blockchain is only applied as an additional feature but not as part of the core infrastructure e.g. to store reputation [36], [37], [38], [39] or as a way to randomly choose an aggregator out of the client pool [40], [41].

While [46] acknowledges multiple factors like data quality and task satisfaction as affecting rewards, it remains vague about its contribution measurement methodology. Similarly, [51], [54], and [48] lack specificity in this regard.

2.4.2 Decentralization of FLF

The trade-off in using blockchain lies between scalability and decentralization. Although it is theoretically favorable to decentralize all FL operations - namely Aggregation (A), Coordination (C), Payment (P), and Storage (S) - on-chain, doing so may introduce prohibitive computational and storage costs. This is because all blockchain nodes must replicate both computation and storage at all times. Specifically, the need to store and manipulate data-heavy objects, such as millions of NN parameters, on-chain restricts the framework to a limited number of participants. In summary, Table 1 compares decentralized and incentivized FLF to the approach presented in this work. Our approach is unique in allowing for NN flexibility (see Section 2.2) while simultaneously maintaining full decentralization and scalability.

3 Problem Statement and Reward Mechanism

3.1 Problem Statement

We assume a federation $\mathcal{F}$ of workers $\mathcal{W}$ who have a common interest in advancing their private Neural Networks based on (i) additional data from other participants and (ii) the unlabeled public dataset X^{pub} through Federated Distillation (FD). We consider an environment where all participants of $\mathcal{F}$ have equal power. For example, no central entity such as a central server should have the power to either censor or manipulate the reward distribution. Each worker participating in the training is responsible for submitting predictions on the public dataset X^{pub} based on their locally trained model and label distribution $labelCount_i$ of the predictions. To enable decentralization, a smart contract atop a blockchain will replace the central server. This contract will (i) aggregate the workers' predictions and (ii) calculate the rewards considering other contributions. To ensure accountability and to prevent free-riding, each worker must stake a deposit D_i. $\mathcal{D} = \sum_{i \in \mathcal{F}} D_i$ will be used to pay $\bar{\tau}_i$ for each contribution at the end of the training process. Note that $\bar{\tau}_i \geq D_i$ if worker i's contributions are above average to $\mathcal{F}$ and $\bar{\tau}_i \leq D_i$ otherwise. Malicious behaviors, such as (i) withholding after committing and (ii) committing an incorrect label distribution $labelCount_i$, will result in the slashing of the deposit and exclusion from $\mathcal{F}$. The worker selection process is beyond the scope of this work. Reputation systems [55], [56] or required registrations might be feasible solutions. Our proposed framework is designed to be lightweight and blockchain agnostic. By employing 1-bit compressed logits on a public test set, instead of aggregating millions of parameters of modern NN (`FedAvg`), and incorporating a computationally simple, correlation-based reward mechanism, our framework uniquely enables (i) on-chain aggregation and (ii) on-chain reward calculation, while maintaining compatibility with both ASBS and GPBS. While theoretically possible, many promising public blockchain projects are still in their technological infancy, either lacking smart contract functionality or facing scalability restrictions. These constraints currently make deploying our system on public blockchains economically infeasible, due to high transaction fees and limited transactions per second, resulting in scalability issues. Consequently, our framework is specifically designed for the **cross-silo case** on **permissioned blockchains**. We assume the following properties:

1) **Honest Majority Assumption:** We assume that the majority of the nodes in the blockchain network are honest and follow the protocol. This is critical for the blockchain's consensus mechanism to function correctly.
2) **Sybil Attack Resistance:** We assume that our blockchain network is resistant to Sybil attacks, where an adversary controls multiple nodes. This is especially important for the GPBS deployment, where entry to the network is more open.
3) **Confidentiality and Integrity:** We assume that the blockchain ensures the confidentiality and integrity of the data and code.

3.2 Reward Mechanism Motivation

As no entity is in possession of the true labels of X^{pub} in the decentralized Federated Learning setting, workers' evaluations cannot be verified. This might encourage workers to report random data without actually classifying X^{pub}. This can be mitigated by rewarding peer consistency, e.g. the reward depends on its consistency with the label given by other workers. However, the best strategy in such schemes is for all workers to report the same answer without investing effort in finding the real label. The solution to these issues is to set up a mechanism, where the expected profit for each individual worker is maximized, if they put high effort into solving the task while acting truthful. In contrast to a server-worker relationship, our framework assumes multiple stakeholders with common interest in improving their respective model. The initially staked deposit D which will be used to pay τ manifests this mutual interest. Yet, contributions may be of different quality to the overall federation. Low quality workers may even have a negative effect on the overall federation even if their intention is truthful. At the same time, some classes in X^{pub} may be less common and therefore are more important to classify correctly. Hence, a mechanism is required to:

1) incentivize only workers with the best abilities for the task
2) incentivize these workers to invest their utmost effort in obtaining the most accurate answer
3) incentivize workers who are able to classify uncommon samples in X^{pub} with higher rewards

3.3 Peer Truth Serum for Federated Distillation

The Peer Truth Serum for Crowdsourcing (PTSC) is a promising Multi-task Peer Prediction mechanism. Through a scoring rule τ, it rewards workers for surprisingly

Ref.	FL	Type	NN-flex.	A	C	P	S	S on BC	CM	TA	Scal.
[45]	FedAvg	n.s.	✗	✓	✗	✓	✓	NN-p	n.s.	✓	limited
[44]	n.s.	n.s.	✗	✓	✗	✓	✓	NN-p, MD	Accuracy, data size	✓(SG)	limited
[52]	FedAvg	CS	✗	✓	✗	✗	✓	NN-p	Generic (Shapley values)	✗	limited
[46]	n.s.	CD	✗	✓	✗	✓	✓	NN-p	n.s.	✗	limited
[41]	FedAvg	CS	✗	✓	✓	✗	✓	NN-p	ED of model updates	✗	limited
[15], [43]	n.s.	n.s.	✗	✓	✓	✓	✓	NN-p, MD	Accuracy (generic)	✓(CT)	limited
[42]	n.s.	CD	✗	✗	✓	✗	✓	NN-p	Accuracy (loss)	✓	limited
[36]	FedAvg	n.s.	✗	✗	✓	✓	✓	NN-p	ED of model updates	✗	limited
[54]	n.s.	n.s.	✗	✗	✗	✗	✓	NN-p	Accuracy (loss)	✗	limited
[51]	n.s.	n.s.	✗	✗	✗	✗	✓	NN-p	n.s.	✓	limited
[50]	n.s.	n.s.	✗	✗	✗	✗	✓	NN-p,MD	Data size, sensing capacity	✓(RL)	limited
[49]	n.s.	n.s.	✗	✗	✗	✓	✓	NN-p	Accuracy	✓	limited
[48]	n.s.	n.s.	✗	✗	✗	✓	✓	NN-p	n.s.	✗	limited
[47]	FedAvg	n.s.	✗	✗	✓	✓	✓	NN-p,MD	Accuracy (generic)	✗	limited
[53]	FD	n.s.	✓	✓	✗	✓	✓	n.s.	Similarities to others	✗	(n.s.)
this work	FD	CS,(CD)	✓	✓	✓	✓	✓	logits, MD	PTSFD	✓	good

TABLE 1: Summary of decentral and incentivized Federated Learning Frameworks. *(Ref. = Reference, NN-flex. = Neural Network flexibility, A = Aggregation, C = Coordination, P = Payment, S = Storage, S on BC = Storage on Blockchain, CM = Contribution measurement, TA = Theoretical analysis, Scal. = Scalability, MD = Metadata, n.s. = Not specified, CS = Cross-silo, CD = Cross-device, NN-p = Neural Network parameters (e.g. gradients, models, model updates), SG = Stackelberg game, CT = Contract theory, ED = Euclidean distance, RL = Reinforcement Learning)*

common reports, encouraging honest and high-effort behavior without the need for ground-truth knowledge [16]. PTSC merges the reward mechanism of [57] with the Peer Truth Serum concept [58], [59], ensuring incentive compatibility across a non-binary solution space suitable for heterogeneous workers. Introducing the Peer Truth Serum for Federated Distillation (PTSFD), we adopt the PTSC framework, as described in Algorithm 1, for the Federated Distillation setting. In this scenario, a group of workers perform statistically independent tasks, where a task refers to classifying a sample j, with $j \in X^{pub}$. The discrete density function is represented as:

Algorithm 1: The Peer Truth Serum for Crowdsourcing [16]

```
1 Step 1
2     CalculateFrequency(all tasks except worker i's report)
3     Let $R_i(x) = \frac{\text{num}(x)}{\sum_y \text{num}(y)}$ be the frequency of
          reports (excluding worker $i$), where $num$
          counts the occurrences of reported values $x$
4 Step 2 SelectPeerWorker(task j)
5     Select the peer worker $p$ for task $j$
6 Step 3 RewardForReporting($x_w$)
7     Worker $i$ is rewarded for reporting $x_{ij}$ on task
          $j$ with the score:
              $\tau(x_{ij}, x_{pj}) = \lambda \cdot (\tau_0(x_{ij}, x_{pj}) - 1)$
          where $x_{pj}$ is the report from worker $p$ on task
          $j$, and $\lambda > 0$. The function $\tau_0$ is given by:
              $\tau_0(x_{ij}, x_{pj}) = \begin{cases} \frac{1}{R_i(x_{ij})} & \text{if } x_{ij} = x_{pj} \\ 1 & \text{if } x_{ij} \neq x_{pj} \end{cases}$
```

$$R_i(x) : \mathcal{C} \mapsto [0,1], \sum_{x \in \mathcal{C}} R_i(x) = 1 \tag{2}$$

Here, $R_i(x)$ excludes the contribution from worker i and denotes the fraction of reported labels, given by:

$$R_i(x) = \frac{\text{labelCount}(x)}{\sum_{y \in \mathcal{C}} \text{labelCount}(y)} \tag{3}$$

Furthermore, PTSFD incorporates an adjustable penalty term β. This modification acknowledges that the primary motivation might be the utility of an improved model, making the payment secondary (as seen in Equation 6). Consequently, the reward for each sample is:

$$\tau_{ij}(x_{ij}) = \lambda \cdot \left(\frac{1}{n_{\text{peers}}} \sum_p \tau_0(x_{ij}, x_{pj}) - \beta \right) \tag{4}$$

Where λ adjusts the payment magnitude and β modulates the reward-accuracy ratio. The cumulative reward for worker i is computed over all tasks as:

$$\bar{\tau}_i = \sum_{j \in X^{pub}} \tau_{ij} \tag{5}$$

3.4 Game-theoretic Analysis

The setting can be considered a two-stage game. In stage 1, workers choose the amount of effort e they want to invest in classifying X^{pub}. To simplify the analysis, we assume two levels of effort, high e_1 and low e_0. Here, e_1 represents the best work possible exerted by the worker, and e_0 represents no effort (i.e., no local NN training). Unlike in FedAvg-based systems, the proposed framework applies FD, hence it does not require a uniform NN among the clients but allows for flexible architectures appropriate for the hardware constraints of the respective workers (see Section 2.2). In stage 2, workers decide on what to report. The baseline model assumes each worker solves every task.

Yet, without loss of generality, workers could be randomly allocated to solve tasks such that each sample of X^{pub} is classified by at least two different workers.

Workers. We assume workers to be individually rational, aiming to maximize their expected profit $\Pi_i = \text{Rewards}_i - \text{Costs}_i$:

$$\max \mathbb{E}(\Pi_i) = U_i(\theta_i^{improved}) + U_i\left(\bar{\tau}_i - [c_i(e_i) + c_{\mathcal{S}}^{fix}]\right) \quad (6)$$

U_i represents the expected utility function of worker i, which can vary among workers. The expected rewards of contributing to the federation $\mathcal{F}$ are twofold: (i) the expected utility of the improved model $\theta_i^{improved}$ and (ii) the utility of the expected monetary reward from $\mathcal{S}$ for contributing to classify X^{pub}. We assume that the training process incurs variable costs $c_i(e_i)$, where c_i is an increasing function of effort e_i. Specifically, $c_i(e_1) > c_i(e_0)$, where e_0 denotes no effort and e_1 denotes high effort of worker i. Effort represents the quality and quantity of private data, model quality, number of training iterations, etc. A detailed Pareto-optimal cost analysis [60], [61] under real-world assumptions will be explored in future work. Additionally, to offset free-riding of inactive but registered workers of $\mathcal{S}$ who benefit from an improved model $U_i(\theta_i^{improved})$ without contributing, fixed participation costs $c_{\mathcal{S}}^{fix}$ are necessary. The initially staked deposit is used to pay contributing workers. Thus, $c_{\mathcal{S}}^{fix} = D_i^{before} - D_i^{after}$ describes the implicit costs for accessing Y_{aggr}^{pub}.

Incentive Compatibility. In order to evaluate PTSFD in game theoretic terms, we analyze each workers expected profit $\Pi_i = Rewards_i - Costs_i$. We assume *Individual Rationality* (IR), e.g. workers try to maximize their expected profit and do not participate if $\Pi \leq 0$. For the sake of simplicity, we further assume that the gain in model improvement $U_i(\theta_i^{improved})$ is offset by $U_i(c_{\mathcal{C}}^{fix})$. When a worker classifies a sample, it obtains an evaluation Y_j^{eval} which can be different from the reported value Y_j^{report}. In stage two, workers face three different strategies $\forall j \in X^{pub}$ [16]:

1) **Honest** Invest high effort e_1 to obtain Y_j^{eval} and report honestly, s.t. $Y_j^{report} = Y_j^{eval}$
2) **Strategic** Invest high effort e_1 to obtain Y_j^{eval} but reports $Y_j^{report} \neq Y_j^{eval}$
3) **Heuristic** Do not invest any effort e_0 and randomly report Y_j^{report} based on the a-priori known distribution of labels in X^{pub}

We define the mechanism to be incentive compatible, if the honest strategy is the dominant strategy for every worker. We use an equilibrium analysis to determine the resulting behavior of each worker. In particularly, $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_n)$ represents a strategy profile of each worker. This profile is an equilibrium $\bar{\sigma}$ if for any worker $i \in \mathcal{W}$, the workers expected profit is maximized with the honest strategy profile $\bar{\sigma}$. Suppose that worker i believes that the peer workers are honest and their answer on a given sample j is positively correlated with the worker i's answer x, when obtained with high effort e_1. Specifically, worker i believes that answer x is not less likely for sample j than in the distribution over all tasks.

Honest Strategy. For every sample j in X^{pub}, the worker calculates the probability scores over all possible classes in $\mathcal{C}$ (output of the softmax layer of a NN). Let us further assume worker i is in possession of a trained model θ_i, with an overall accuracy $Accuracy_{\theta_i}$. We define the relative certainty $\mathbb{A}_{ij}$ of any prediction of client i on an element j of X^{pub} as the product of the local classifier accuracy and the sample-specific maxprobabilityscore.

$$\mathbb{A}_{ij} = Accuracy_{\theta_i} \cdot MaxProbabilityScore_{ij} \quad (7)$$

Under the assumption that the local client data X_i^{priv} is representative of the entire data distribution D, this metric will give a heuristic measure for the data specific certainty in the model prediction. Based on this metric, each worker will make the decision whether to report predicted labels, discarding those for which reward is expected to be negative. This leads to the expected profit

$$\mathbb{E}(\Pi_{ij}) = \mathbb{A}_{ij} \cdot \lambda\left(\frac{1}{R(x_{ij})} - \beta\right) + (1 - \mathbb{A}_{ij}) \cdot \lambda(-\beta) - c_i(e_i) \quad (8)$$

Assuming individual rationality, $\mathbb{E}(\Pi_{i,j}) \geq 0$ in order to incentive worker i to submit a vote on sample j. Following 8, we can derive minimum prediction quality

$$\mathbb{A}_{ij} \geq R(x_{ij}) \cdot \left(\frac{c_i(e_i)}{\lambda} + \beta\right) \quad (9)$$

required to incentivize worker i to participate, e.g. $\Pi_i \geq 0$. Notice that the federation can set the overall quality threshold by adjusting hyperparameter λ and β appropriately, assuming similar variable costs $c(e)$ on the workers side.

Heuristic Strategy. The heuristic strategy assumes that worker i does not exert any effort to obtain $Y_j^{eval} = x \in \mathcal{C}$. The expected reward is based on the probability of matching a peer's answer. Given that answer x is independent of the task, the probability of coincidentally matching a peer is equivalent to the frequency of answer $x \in \mathcal{C}$.

$$\mathbb{E}(\Pi_{ij}) = R(x_{ij}) \cdot \left(\frac{1}{R(x_{ij})} - \beta\right) + (1 - R(x_{ij})) \cdot (-\beta) = 1 - \beta \quad (10)$$

It's important to note that the expected profit for $\beta = 1$ is 0, and it is strictly negative for $\beta > 1$. This holds irrespective of what answer x is, or what the worker knows about the distribution $R(x)$ across labels in X^{pub}. Given that the noise introduced to classify X^{pub} diminishes the overall model quality, as per Equation 8, a rational worker is unlikely to participate if $\beta \geq 1$.

Strategic Strategy. Assuming the honest participation of other workers, exerting Y_j^{eval} while reporting $Y_j^{report} \neq Y_j^{eval}$ consistently results in a negative expected profit for all $j \in X^{pub}$.

$$\mathbb{E}(\Pi_{ij}) = \lambda \cdot (-\beta) - c_i(e_i) \quad (11)$$

This is true provided the self-predicting condition [16] is met, i.e.,

$$\frac{A_{ij}(x|x)}{R(x)} > \frac{A_{ij}(\bar{x}|x)}{R(\bar{x})}, \forall \bar{x} \neq x \quad (12)$$

Considering a scenario where workers collude (they report

x for both $Y_j^{eval} = x$ and $Y_j^{eval} = y$), R will adjust such that $R^{col}(x) = R(x) + R(y)$.

$$\mathbb{E}(\Pi_{ij}) = \begin{cases} \mathbb{A}_{ij} \cdot \lambda \left(\frac{1}{R_i(x_{ij}) + R_i(y_{ij})} - \beta \right) \\ +(1 - \mathbb{A}_{ij}) \cdot \lambda(-\beta) - c_i(e_i) & \text{if } Y_{ij}^{eval} = x_{ij} \\ \mathbb{A}_{ij} \cdot \lambda \left(\frac{1}{R_i(x_{ij}) + R_i(y_{ij})} - \beta \right) \\ +(1 - \mathbb{A}_{ij}) \cdot \lambda(-\beta) - c_i(e_i) & \text{if } Y_{ij}^{eval} = y_{ij} \end{cases} \tag{13}$$

This change in R entirely offsets the increase in the probability of a match. Therefore, only an honest strategy paired with a high-quality model will yield a positive expected reward for a given worker. This results in an equilibrium $\bar{\sigma}^{honest}$ for the PTSFD mechanism, thereby demonstrating its incentive compatibility.

4 1-bit Compressed Federated Distillation Framework with Smart Contract Logic

The protocol consists of the following steps: (i) Task Specification & Contract Deployment, (ii) Worker Registration & Deposit, (iii) Local Model Training, (iv) Hash Commit Submission, (v) Reveal Predictions, (vi) Aggregation & Reward Distribution, and (vii) Knowledge Distillation from X^{pub}.

4.1 Task Specification & Smart Contract Deployment

To form Federation $\mathcal{F}$, participants with similar interests must agree on the requirements and specifics of an FD task, specifically:

1) Task description and data distribution (e.g., images of a certain type).
2) Reference to a public data set X^{pub} and potential classes $\mathcal{C}$ for the Federated Distillation pipeline. This will subsequently be utilized by workers to predict the labels on each sample of the dataset.
3) Reference to the address of $\mathcal{S}$.
4) Deposit amount D_i that each worker must stake.
5) PTSFD and reward mechanism details (λ and β values).

After forming a federation $\mathcal{F}$, either an external third party or one of the workers from $\mathcal{F}$ deploys the governing smart contract $\mathcal{S}$, stakes the necessary deposit D_i, and lists the addresses of all eligible workers in $\mathcal{F}$, as well as the aggregation and PTSFD logic of the FD task.

4.2 Worker Registration & Deposit Submission

Based on the task specifications, interested workers register on the smart contract $\mathcal{S}$ using their respective blockchain address (public key) and submit the required deposit D_i. $\mathcal{S}$ verifies if the applying worker belongs to the federation. Given that $|\mathcal{F}| >> |\mathcal{W}|$, PTSFD motivates valuable workers for $\mathcal{F}$ in terms of data and computational capacity to engage while dissuading low-quality workers, as demonstrated in Section 4.5. To preclude free-riding, workers in $\mathcal{F}$ who aren't registered shouldn't access $\mathcal{S}$. This restriction can be implemented by deploying $\mathcal{S}$ on a suitable blockchain system or by shuffling X^{pub}, ensuring only registered clients can access the correct indices.

4.3 Local Model Training and Prediction

The entire training procedure encompasses two phases: the local model training phase on local data X_i^{priv}, Y_i^{priv} and the KD phase from X^{pub}, Y_{aggr}^{pub}, which occurs as the protocol's final step, as detailed in Section 2.2.

Training on Local Data. Each worker either has a pre-trained model or begins training a NN on their specific private data until convergence (optionally, until a predetermined minimum accuracy agreed upon within $\mathcal{F}$ is achieved). Notably, unlike `FedAvg`, FD doesn't mandate a common shared NN architecture across all workers, thus enabling them to select an optimal architecture tailored to their computational resources.

Label Prediction. Upon completing the training, workers compute the soft labels $Y_i^{pub} = \{f_{\theta_i + \Delta\theta_i}(x) \mid x \in X^{pub}\}$ and subsequently quantize these to 1-bit as $Y_i^{1bit} = Q_{1bit}(Y_i^{pub})$.

Label Count. The PTSFD mechanism necessitates data on the label distribution $R(x)$ over X^{pub} for reward calculations. Hence, each worker i must compute the label count $labelCount_i \in \mathbb{N}^{|\mathcal{C}|}$ for each label present in X^{pub}, to minimize computational overhead on the blockchain (as described in Algorithm 2). The supplemental validation function to ensure the accurate computation of $labelCount_i$ depends on the specific blockchain system and is outside this work's purview.

Algorithm 2: Local label count for worker i

```
input : Integer encoded class votes $x_{ij}$, where
        $i \in \mathcal{W}' \subseteq \mathcal{W}$
output: $labelCount_i$
1 init $labelCount_i$
2     var $labelCount_i \in \mathbb{N}^{|\mathcal{C}|} = (0, 0, \ldots, 0)$
3 foreach $j \in X^{pub}$ do    // iterate over data samples
4     $labelCount_i(x_{ij})$ += 1
5 return $labelCount_i$
```

4.4 Commit and Reveal

Information on the blockchain is transparent to every node. Even in a private blockchain setup, workers in $\mathcal{W}$ could wait for peers to publish Y_p^{1bit} and replicate their results without expending any effort. To prevent this kind of copying and to ensure that workers apply effort to classify X^{pub}, a two-step commit and reveal scheme is employed.

Commit. Prior to publishing the results to $\mathcal{S}$, where all peer workers could view the submission, a cryptographic hash $hashCommit_i = \mathcal{H}\left(Y_i^{1bit}, salt_i, labelCount_i\right)$ is computed to obfuscate Y_i^{1bit} and $labelCount_i$. The property of pre-image resistance of a cryptographic hash function (e.g., it should be challenging to find any message m such that $commit_i = \mathcal{H}(m)$) and the property of collision resistance (e.g., it should be challenging to find two distinct messages $m1$ and $m2$ with $\mathcal{H}(m_1) = \mathcal{H}(m_2)$) ensure that

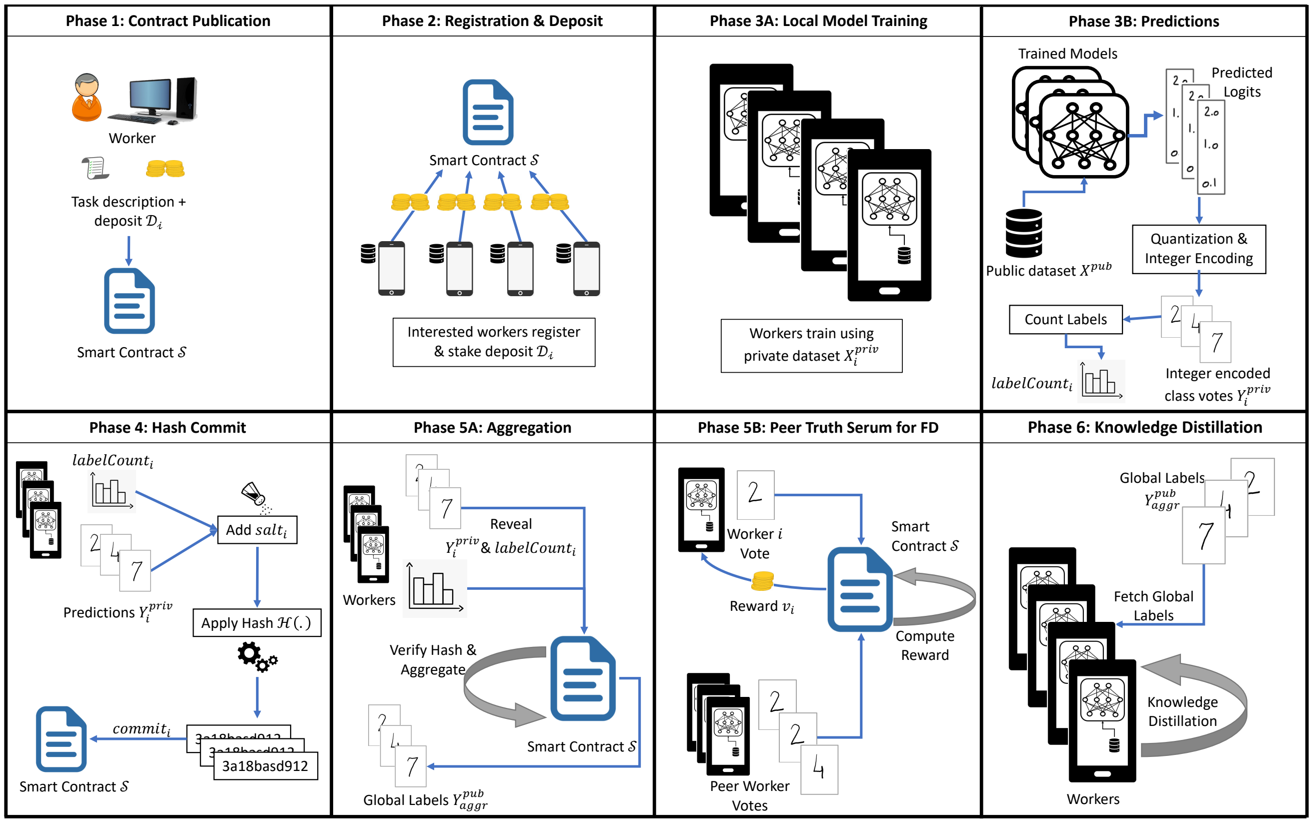
Phase 1: Contract Publication
Worker
Task description + deposit $\mathcal{D}_i$
Smart Contract $\mathcal{S}$
Phase 2: Registration & Deposit
Smart Contract $\mathcal{S}$
Interested workers register & stake deposit $\mathcal{D}_i$
Phase 3A: Local Model Training
Workers train using private dataset X_i^{priv}
Phase 3B: Predictions
Trained Models
Predicted Logits
Public dataset X^{pub}
Quantization & Integer Encoding
Count Labels
$labelCount_i$
Integer encoded class votes Y_i^{priv}
Phase 4: Hash Commit
$labelCount_i$
Add $salt_i$
Predictions Y_i^{priv}
Apply Hash $\mathcal{H}(.)$
$commit_i$
3a18basd912
Smart Contract $\mathcal{S}$
Phase 5A: Aggregation
Reveal Y_i^{priv} & $labelCount_i$
Workers
Verify Hash & Aggregate
Smart Contract $\mathcal{S}$
Global Labels Y_{aggr}^{pub}
Phase 5B: Peer Truth Serum for FD
Worker i Vote
Smart Contract $\mathcal{S}$
Reward v_i
Compute Reward
Peer Worker Votes
Phase 6: Knowledge Distillation
Global Labels Y_{aggr}^{pub}
Fetch Global Labels
Knowledge Distillation
Workers

Fig. 3: Iterative Process of on-blockchain Federated Distillation.

no worker can either retrieve Y_i^{1bit} or alter their previously committed Y_i^{1bit}. Each worker i transmits $hashCommit_i$ to $\mathcal{S}$ upon completing their training. It's noteworthy that the commit phase on $\mathcal{S}$ concludes once $|\mathcal{W}'| \subseteq |\mathcal{W}|$ workers have registered with $\mathcal{S}$ or when the maximum time T_{commit}^{max} is reached.

Reveal. During the reveal phase on $\mathcal{S}$, each worker that successfully committed in the commit phase must disclose Y_i^{1bit}, $labelCount_i$, and $salt_i$ within the timeframe T_{reveal}^{max} via a transaction function call to $\mathcal{S}$. To counteract withholding attacks, a worker's deposit D_i is forfeited if worker i fails to reveal within the allotted time T_{reveal}^{max}. The smart contract then verifies the commitment's validity to confirm that $\mathcal{H}\left(Y_i, labelCount_i, salt_i\right) == hashCommit_i$. Algorithm 3 provides the pseudocode for this scheme in Solidity on the Ethereum blockchain.

4.5 Aggregation & Reward Distribution

We apply PTSFD to calculate the reward distribution for each worker. In order to calculate the rewards, $\mathcal{S}$ aggregates $labelCount_i$ across all workers $i \in \mathcal{W}'$ first to obtain the global label count $G = \sum_{\mathcal{W}'} labelCount_i \in \mathbb{N}^{|\mathcal{C}|}$. G is a helper variable to calculate R_i:

$$R_i = \frac{1}{m \times n} \times (G - labelCount_i) \tag{14}$$

The worker is rewarded for its prediction on sample j with respect to it's peers regarding Equation 4. The final *rewardScore* for worker i is a sum of all individual rewards over X^{pub}, given by

Algorithm 3: Commit and Reveal Protocol

```
Data: 32byte $hashCommit_i \leftarrow \mathcal{H}\left(Y_i^{1bit}, labelCount_i, salt_i\right)$
1  Init
2      var commitments ← Mapping($address_i \rightarrow$ byte32) $\forall i \in \mathcal{W}$
3      var userIsCommitted ← Mapping($address_i \rightarrow$ bool) $\forall i \in \mathcal{W}$
4  Phase I commit ($hashCommit$)
5      foreach $i \in \mathcal{W}'$ do
6          require ($userIsRegistered(msg.sender)$) // registered in $\mathcal{W}$
7          require ($!userIsCommited(msg.sender)$)
8          $commitments.append(commit_i)$
9          $isCommitted(msg.sender) = True$
10 Phase II reveal ($Y_i^{1bit}$, $salt$)
11     foreach $i \in \mathcal{W}'$ do
12         require ($userIsCommitted(msg.sender)$)
13         require ($\mathcal{H}\left(Y_i^{1bit}, labelCount_i, salt_i\right) == commitments(msg.sender)$)
```

$$\begin{aligned}\bar{\tau}_i &= rewardScore(i) \\ &= \lambda \cdot \left(\frac{1}{n_j^{peers}} \sum_j \sum_p \tau_0(x_{ij}, x_{pj})\right) \forall i \in \mathcal{W}'\end{aligned} \tag{15}$$

where parameter λ describes a scaling parameter for the reward and n_j^{peers} describes the number of peer workers who also submitted a label prediction on j. The aggregated predictions Y_{aggr}^{pub} are calculated by majority vote of $Y_i^{1bit}\ \forall i \in \mathcal{W}'$. We merge the reward computation and aggregation into a single algorithm outlined in Supplementary Materials B. Note that implementation details may differ fundamentally depending on the underlying blockchain architecture.

4.6 Knowledge Distillation on Public Dataset

Finally, workers download the aggregated predictions Y_{aggr}^{pub} from the blockchain and perform several epochs of KD using X^{pub} and Y_{aggr}^{pub} to improve their respective models ($\theta_i^{improved} \rightarrow \theta_i + \Delta\theta_i$). Optionally, the training process of each client, as shown in Section 2.2.2, can be repeated until a specific threshold is achieved as specified in the Smart Contract $\mathcal{S}$. Note that λ should decrease with every consecutive round, as most evaluated labels will not change.

4.7 Complexity Analysis

Because we are executing this protocol on the blockchain, it's essential to understand the computational and storage costs involved. In this section, we discuss the overheads related to computation and storage that our proposed algorithm introduces. While the actual implementation on a general-purpose blockchain system might differ based on the underlying virtual machine, our PTSFD implementation, as illustrated in Supplementary Materials B, provides a useful reference to estimate the complexity.

4.7.1 Computational Complexity

The algorithm presented in Supplementary Materials B first calculates the global label distribution and counts class votes across all workers (*lines 7 - 12*). This computational overhead is $\mathcal{O}(m \cdot n)$, where $m = |X^{pub}|$ and $n = |\mathcal{W}'|$. Subsequently, we examine each data sample in X^{pub}, rewarding or penalizing a worker based on its peers. We also determine the aggregated class label for each sample during this algorithm phase (*lines 13 - 29*). The process of calculating the reward for each worker based on its peers results in a computational overhead of $\mathcal{O}\left(m \cdot \sum_{i=1}^{n} n_{peers}\right)$. The global label calculation adds a cost of $\mathcal{O}(m \cdot |\mathcal{C}|)$. In the baseline scenario, where each worker processes all data samples from the public dataset and is considered a peer of all other workers, the overall computational cost is given by Equation 16.

$$\mathcal{O}(m \cdot (n^2 + |\mathcal{C}|)) \tag{16}$$

For more efficient solutions, we distribute public dataset samples among workers such that each sample is classified by a maximum of two workers. Implementing PTSFD in this manner would reduce the overhead as detailed in Equation 17.

$$\mathcal{O}(m \cdot (2n + |\mathcal{C}|)) \tag{17}$$

4.7.2 Storage Complexity

Two types of storage costs are associated with the proposed algorithm: permanent storage and temporary memory variables. $Votes$, M, S, R_i, and τ_0 need memory storage during the computation, resulting in $\mathcal{O}(|\mathcal{C}| \cdot (m+2) + n)$ additional memory storage. Whether the reported frequencies $labelCount_i$ or each worker's final reward share $rewardScore$ need permanent blockchain storage depends on the requirements of the specific blockchain system. Ideally, only $globalLabels = Y_{aggr}^{pub}$ is stored permanently on the blockchain. The minimum amount of data required for each round is represented by Equation 18, where η accounts for the overhead due to encoding necessities.

$$\mathrm{b}_{globalLabels} = n \times |\mathcal{C}| \times 1\mathrm{bit} + \eta \tag{18}$$

4.8 PTSFD in Comparison to Shapley Value

Quantifying individual contributions in FL is essential for the equitable distribution of rewards and the growth of FL systems. The Shapley value is a concept in cooperative game theory that distributes total gains among players by measuring the marginal contributions each player makes to different possible coalitions. For a worker i in a set $\mathcal{W}$, with utility evaluation function $V : 2^{\mathcal{W}} \rightarrow \mathbb{R}$, the Shapley value $\phi_i(V)$ is given by:

$$\phi_i(V) = \frac{1}{|\mathcal{W}|!} \sum_{\Pi} \left[V\left(S_i^{\Pi} \cup \{i\}\right) - V\left(S_i^{\Pi}\right) \right], \tag{19}$$

where Π is the set of all permutations of $\mathcal{W}$, and $S_i^{\Pi} \subset \mathcal{W}$ is the set of workers preceding i in permutation Π. The Shapley value adheres to several axioms for fairness:

1) **Efficiency:** Total utility is fully distributed among the workers: $\sum_{i \in \mathcal{W}} \phi_i(V) = V(\mathcal{W})$.
2) **Symmetry:** Workers contributing identically to every subset receive identical rewards.
3) **Null Player:** Workers who do not enhance utility for any coalition yield no reward.
4) **Additivity:** The Shapley value is linear over the utility functions of combined games.

Despite these favorable properties, the computation of the Shapley value scales with $O(2^n)$. In FL, $V(S) = V(\theta_S)$, where θ_S is the FL model trained on the subset of datasets $\{X, Y\}_S = \{X_i, Y_i\}, i \in S$ from scratch for every permutation:

$$V(S) = V(\theta_S) = V(A(\theta^{init}, \{X, Y\}_S)) \tag{20}$$

where $A(\cdot)$ is a learning algorithm and θ^{init} denotes the initial model. The computational demands involved in calculating the Shapley Value render it challenging to efficiently execute even a single step on a general-purpose blockchain. Recent studies have attempted to approximate the Shapley value to mitigate this computational overhead [62], [63], [64].In contrast, PTSFD offers a lightweight method to elicit truthful contributions implicitly, bypassing the exhaustive computation of the marginal utilities. PTSFD, as an alternative, is designed to be compatible with blockchain technology, enabling scalable and decentralized FL without excessive computation, while still aspiring to maintain fairness in incentives.

4.9 Limitations

Despite the benefits of our decentralized FD protocol, our framework encounters the following limitations.

Public Dataset. Although FD offers numerous advantages, such as reduced information exchange and the flexibility of independent NN architectures, the FD training process necessitates access to a public dataset X^{pub}. This may not be available for certain use cases. While [65] demonstrated that highly disparate data distributions might suffice for FD, relying on $\mathbb{A}_{ij}$ as a heuristic for the evaluation certainty of a sample limits the variance of distributions between X^{pub} and X^{priv}. The scoring method proposed by [66] appears promising in addressing this.

Public Blockchains. Despite significantly reducing computational and storage demands, our framework remains unsuitable for current public blockchain systems due to (i) the high costs associated with storing Y^{pub}_{aggr} and the computational overhead of PTSFD, and (ii) the transparency of Y^{pub}_{aggr} to nodes that are not members of $\mathcal{W} \in \mathcal{F}$ and hence did not make a deposit. Both issues may be addressed by upcoming advancements in the public blockchain sector.

Self Predicting Condition. PTSFD is incentive-compatible and yields an optimal outcome when workers are honest. However, the mechanism's incentive compatibility is contingent upon the satisfaction of Equation 12. If classes are evenly distributed across X^{pub}, the conditions will always be met. Example: Let $Pr(x = a) = 0.8$ and $Pr(x = b) = 0.2$, but $R(a) = 0.9$ and $R(b) = 0.1$. Although the worker's $Y_i^{eval} = a$, their expected reward would be greater if $Y_i^{report} = b$, as $\frac{0.8}{0.9} - \beta < \frac{0.2}{0.1} - \beta$.

5 Experiments

In this section, we empirically evaluate the PTSFD framework and analyze the reward distribution under different levels of effort as well as its robustness in the event of malicious behavior. We do not consider explicit variable costs $c_i(e)$. Furthermore, we set the reward scaling parameter $\lambda = 1$ for all experiments. We do not account for lagging workers, so $\mathcal{W}' = \mathcal{W}$ for all experiments. All experiments are based on multiple rounds of the proposed protocol. The code for our experiments is made publicly available.[2] Specifically, we experimentally validate the following properties of PTSFD:

1) **Performance:** Choosing to participate in the federation should lead to a significant improvement in model accuracy for each worker.
2) **Fairness:** The greater the effort a worker exerts in terms of training accuracy and amount of training data, the higher the reward they should receive.
3) **Robustness:** Malicious workers should receive substantially less reward, even under high collusion rates.

5.1 Data Sets and Models

We analyze the decentralized 1-bit compressed FD with the PTSFD protocol on three different federated image classification problems, using EMNIST [67] / MNIST [68], CIFAR-10 [69] / STL-10 [70] and Fashion MNIST [71] datasets on ResNet-18 [72] and LeNet [73] and respectively as training/distillation data. Our Federation comprises 4, 10 and 25 workers for different experiments. The training data is distributed among workers according to a Dirichlet distribution with the Dirichlet parameter α. Figure 4 top row illustrates the data distribution of 10 labels across 10 different workers for $\alpha = 100$, $\alpha = 1$, and $\alpha = 0.1$. The different alphas simulate various data distributions such as iid and non-iid. We first train models locally on X^{priv} and then perform KD using the public dataset X^{pub}. Even though in real-world PTSFD applications, workers might train different model architectures and vary the number of local training epochs based on their hardware constraints, we employ a single default NN architecture for simplicity. We simulate heterogeneity through varying local training accuracy (early stopping), non-iid data, and different sizes of X^{pub}. It's important to note that the distribution of the distillation data deviates from the worker's data, mirroring realistic FL scenarios (e.g., MNIST contains handwritten digits, while EMNIST features a different set of handwritten numbers; similarly, CIFAR-10 includes a distinct set of images compared to the STL-10 dataset). We use the Adam optimizer [74] with a fixed learning rate of 0.001 for both the distillation and training processes. We minimize cross-entropy loss for local model training on X^{priv}, Y^{priv} and minimize Kullback-Leibler Divergence on X^{pub}, Y^{pub}_{aggr}.

5.2 Storage and Communication Cost

Given that storage and computation costs on the blockchain are critical scalability constraints for decentralized FLFs [1], we assess our framework's storage costs in comparison to `FedAvg` for specific target accuracies. Table 2 shows the communication cost (upstream/downstream) as well as the storage cost required to achieve a specific accuracy target for ResNet-18 and LeNet on the CIFAR-10, MNIST, and Fashion-MNIST datasets, respectively. All experiments were run under different data distributions by varying the Dirichlet parameter α while simulating 4, 10, and 25 workers respectively. As observed from the summarized results, our framework achieves accuracy similar to that of a typical Federated Learning system using `FedAvg` as the aggregation mechanism but at a fraction of the communication and storage costs. For instance, when training ResNet on CIFAR-10 with $\alpha = 100$, we can achieve the target accuracy with a minimal communication cost of 0.84 MB and storage cost of 0.92 MB, compared to the substantial costs incurred by `FedAvg` (5.33 GB and 5.37 GB for communication and storage costs, respectively, an improvement of roughly 5,000x). The remaining experiments, showcasing model quality improvement, reward fairness, and simulation of collusion or heuristic behavior, were conducted using LeNet on EMNIST/MNIST as training and distillation datasets. We believe that these experiments sufficiently demonstrate the desired results and would yield comparable outcomes for other datasets or models.

5.3 Model Quality Improvement

Figure 4 illustrates the impact of the sizes of the local dataset $|X^{priv}|$ and the public dataset $|X^{pub}|$ on the accuracy for

2. https://github.com/Tsinghua-FL-Team/decentralized-FD

TABLE 2: Communication and storage costs (in MB) for achieving specific accuracy in FL across various datasets, architectures, and data heterogeneity levels. We use 10′000, 12′800, and 5′000 samples respectively for CIFAR-10, MNIST and Fashion-MNIST as distillation data. Communication rounds for target accuracy are in parentheses.

Model	Train / Distill Dataset	Target Accuracy	α	Up/Down	`FedAvg`	1-bit FedDistill
ResNet-18	CIFAR-10/Testset/[STL-10]	0.74 [0.68]	100.0	up/down	5332.50 (5)	0.61 (16)
				store	5375.16 (5)	0.77 (16)
		0.67	1.0	up/down	5332.50 (5)	1.83 (48)
				store	5375.16 (5)	2.29 (48)
		0.50	0.1	up/down	8532.00 (8)	1.22 (32)
				store	8574.66 (8)	1.53 (32)
LeNet	Fashion-MNIST/Testset	0.86	100.0	up/down	78.00 (13)	0.19 (14)
				store	78.24 (13)	0.22 (14)
		0.85	1.0	up/down	78.00 (13)	0.11 (9)
				store	78.24 (13)	0.14 (9)
		0.77	0.1	up/down	72.00 (12)	0.16 (12)
				store	72.24 (12)	0.19 (12)
LeNet	MNIST/EMNIST	0.97	100.0	up/down	36.00 (6)	0.23 (6)
				store	36.24 (6)	0.24 (6)
		0.97	1.0	up/down	36.00 (6)	0.27 (7)
				store	36.24 (6)	0.28 (7)
		0.94	0.1	up/down	54.00 (9)	0.15 (4)
				store	54.24 (9)	0.16 (4)

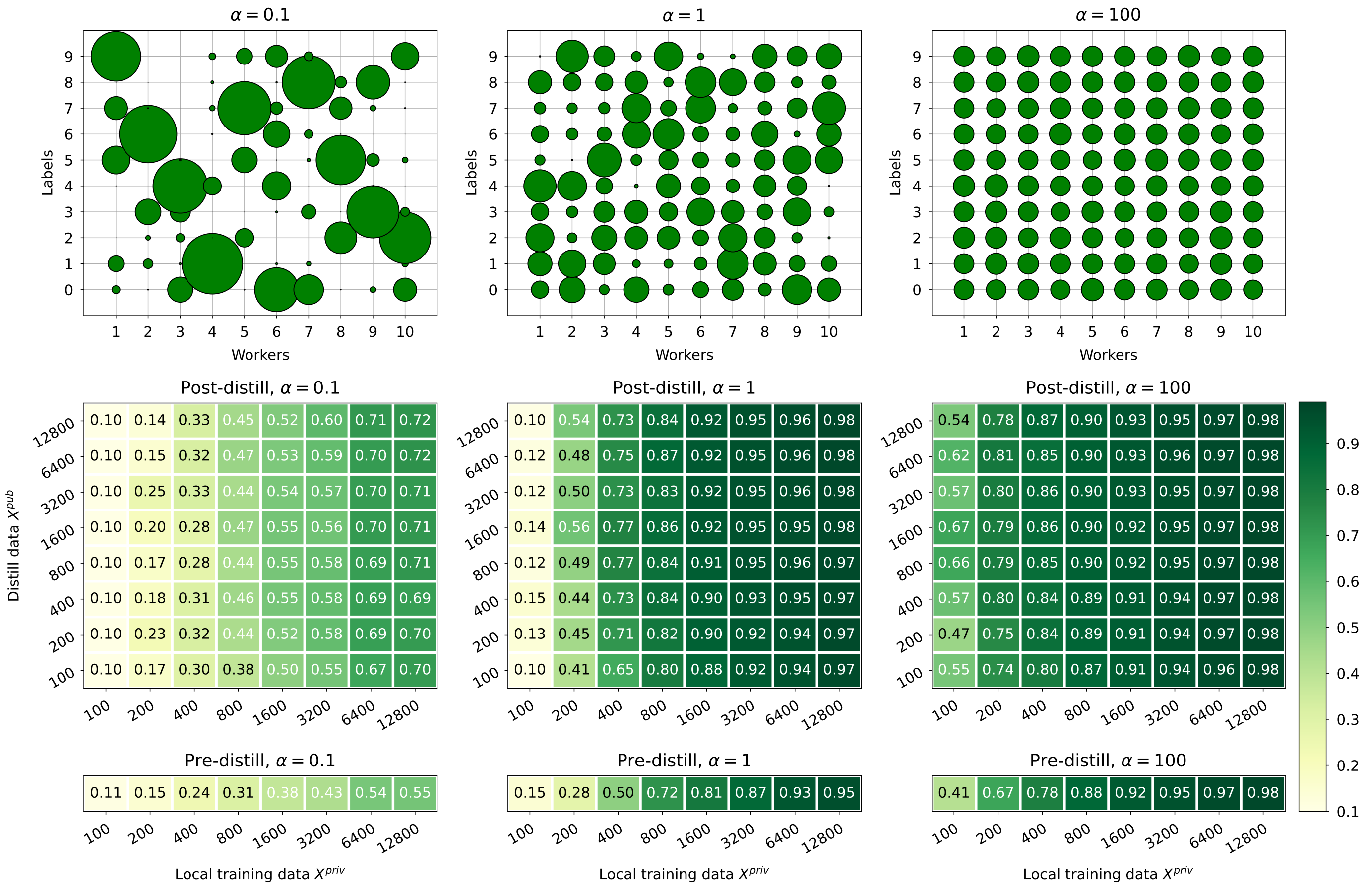

Fig. 4: The influence of X^{pub} and X^{priv} on model accuracy with corresponding dirichlet α setting. These experiments were run using LeNet on MNIST / EMNIST respectively as training / distillation datasets.

EMNIST/MNIST under both non-iid distributions ($\alpha = 0.1$ & $\alpha = 1.0$) and the iid distribution ($\alpha = 100$) for 10 clients. A marked improvement in model quality is evident for every worker following their KD execution. While the size of the local training dataset plays a more substantial role than the distillation dataset, the significance of the latter should not be underestimated, particularly in the context of non-iid distributions which can be attributed to the inclusion of additional data from the public dataset.

5.4 Fair Effort-Reward Correlation

Figure 5 illustrates the correlation between effort, in terms of both heterogeneous training efforts (top) and varying data quantities (bottom), and the subsequent reward distribution. PTSFD facilitates realistic FL scenarios where workers, due to hardware constraints, can employ various local model architectures and train them for different numbers of epochs. For instance, we emulated heterogeneity by training 10 workers using diverse early stopping criteria, where higher local training accuracy, indicative of greater effort, corresponded to better rewards. Concurrently, variations in private data quantity, while assuming consistent data quality, also influence contribution quality. Overall, superior training accuracy and more extensive local dataset result in greater rewards.

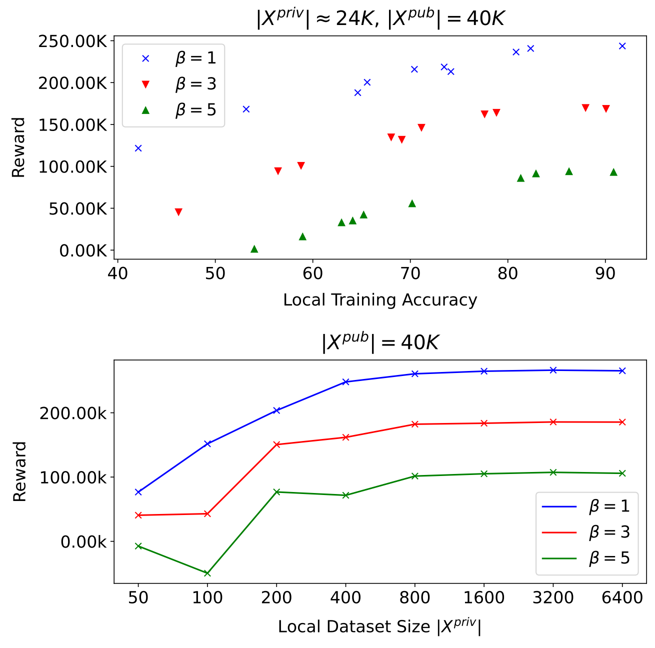

Fig. 5: Effect of local training accuracy & local data size on reward using LeNet on EMNIST.

5.5 Robustness of PTSFD

In order to ensure the desired quality of label predictions, the federation can adjust the parameter λ to scale the reward based on its underlying collateral (with $\lambda = 1$ consistently used in our experiments). Additionally, β can be set to modify the penalty for incorrect answers, thus tuning the confidence threshold required for rationally individual workers (as described in Equation 9) to submit a prediction. The initially staked deposit acts as a safeguard against malicious behavior, as such actions can lead to losses. In our experiment, workers can opt to withhold their reports if they lack confidence in their predictions. Figure 6 demonstrates the reward variations with different penalty factors β across diverse confidence levels. For this experiment, the local training data was divided based on a Dirichlet distribution with parameter $\alpha = 0.1$, mimicking scenarios where workers might not have access to uniform data. As a result, certain workers' local models could be ill-equipped to predict classes previously unavailable to them. These workers will only submit their predictions if their confidence in the most probable label surpasses a certain benchmark. Our findings indicate that, by adjusting β, PTSFD can effectively deter subpar contributions from contaminating the federated training process.

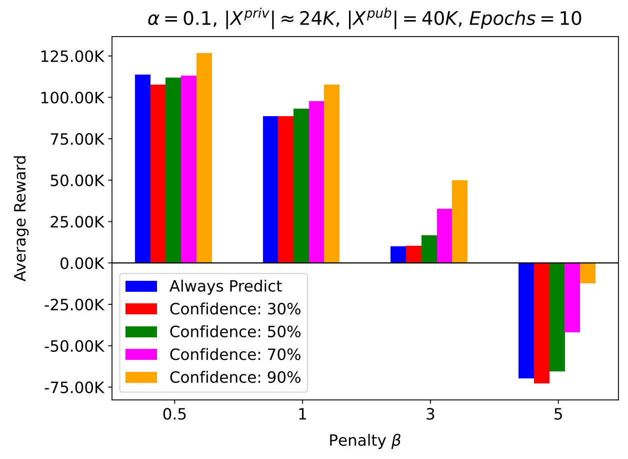

Fig. 6: Effect of confidence based predictions on reward with different β (penalty) using LeNet on MNIST.

5.6 Robustness in Case of Malicious Behavior

Building on the game-theoretic analysis detailed in Section 3.4, we experimentally confirm our theoretical claims in a real FL setting. We have demonstrated that both heuristic behaviors (like bypassing local training to randomly report labels on a public dataset) and strategic tactics like collusion yield an expected reward of $1 - \beta$. Figure 7 (top) depicts the reward differences between colluding and honest workers. Colluding predictions are structured as:

$$Y_i^{report} = \begin{cases} 0 & \text{if } Y_i^{eval} \in \{0,1,2,3,4\} \\ 9 & \text{if } Y_i^{eval} \in \{5,6,7,8,9\} \end{cases}$$

Conversely, Figure 7 (bottom) contrasts the rewards of heuristic workers, who predict randomly on the public dataset, with those of diligent participants. Overall, the data indicate that genuine engagement results in the most significant rewards, even amidst prevalent malicious actions. In-depth cost considerations are reserved for future studies.

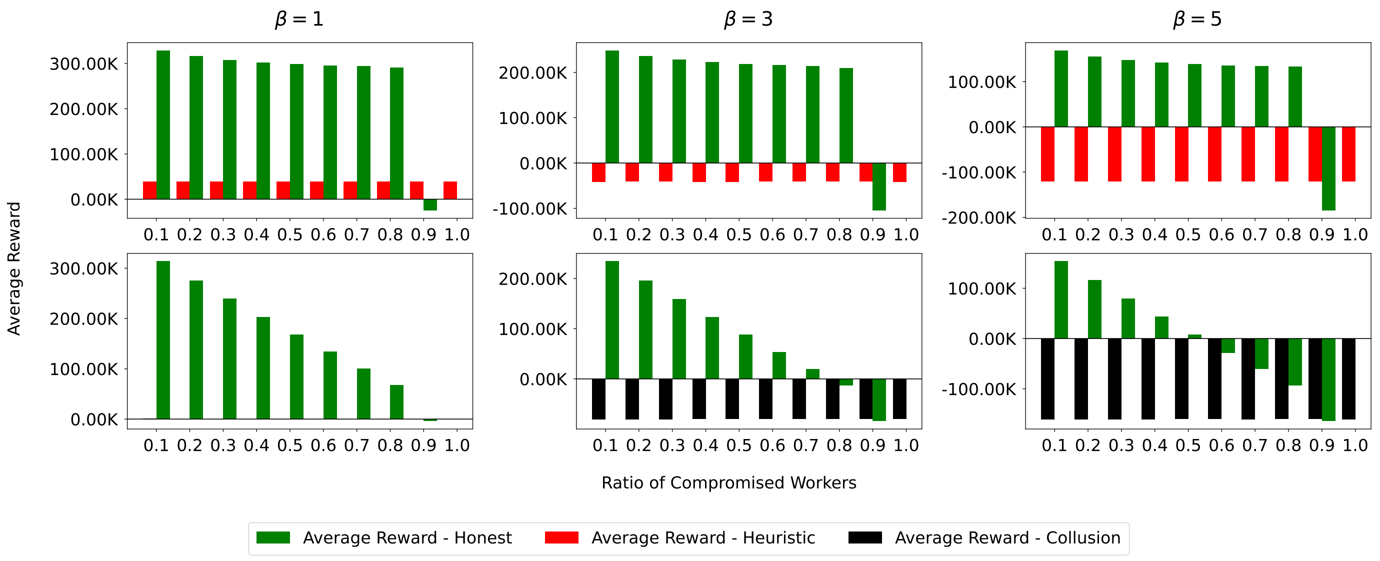

Fig. 7: Average reward with varying ratio of colluding and heuristic workers under different penalty β. Federated Learning setting with 10 workers running LeNet on EMNIST digits for 10 epochs. For all experiments, 40000 data points from the MNIST data set were used as public dataset X^{pub}.

6 Conclusion

In this work, we introduced a novel decentralized and reward-based 1-bit compressed Federated Distillation scheme on the blockchain, incorporating the Peer Truth Serum [16] specifically for Federated Distillation. The 1-bit compression ensures explicit comparability between contributions, a critical feature for automatically computing rewards on a smart contract atop a general-purpose blockchain system, where each worker is regarded as an equitable member of the federation. We have demonstrated that, in terms of storage on the blockchain and communication overhead, our framework is significantly more efficient than Federated Averaging. Additionally, the system not only offers flexibility in neural network architecture but also allows adaptation to various thresholds of contribution quality by adjusting the penalty term β. Furthermore, both theoretical insights and experimental evidence suggest our proposed mechanism is resilient to random reporting and collusion. We are confident that our findings will further the scalability of Federated Learning tasks in fully decentralized environments, where all entities have an equal interest in enhancing their models.

References

[1] L. Witt, M. Heyer, K. Toyoda, W. Samek, and D. Li, "Decentral and incentivized federated learning frameworks: A systematic literature review," *IEEE Internet of Things Journal*, vol. 10, no. 4, pp. 3642–3663, 2023.
[2] H. Brendan McMahan, E. Moore, D. Ramage, S. Hampson, and B. Agüera y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017*, 2017.
[3] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger."
[4] D. G. Wood, "Polkadot: Vision for a heterogeneous multi-chain framework."
[5] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro *et al.*, "Hyperledger fabric: A distributed operating system for permissioned blockchains," in *Proceedings of the Thirteenth EuroSys Conference*, ser. EuroSys '18, 2018.
[6] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet *et al.*, "Advances and open problems in federated learning," *arXiv:1912.04977*, 2019.
[7] T. Nishio, R. Shinkuma, and N. B. Mandayam, "Estimation of individual device contributions for incentivizing federated learning," in *2020 IEEE Globecom Workshops (GC Wkshps*, 2020, pp. 1–6.
[8] R. D. Cook, "Detection of Influential Observation in Linear Regression," *Technometrics*, vol. 19, no. 1, pp. 15–18, apr 1977. [Online]. Available: http://www.jstor.org/stable/1268249
[9] X. Tu, K. Zhu, N. C. Luong, D. Niyato, Y. Zhang, and J. Li, "Incentive mechanisms for federated learning: From economic and game theoretic perspective," 2021.
[10] R. Jia, D. Dao, B. Wang, F. A. Hubis, N. Hynes, N. M. Gurel, B. Li, C. Zhang, D. Song, and C. Spanos, "Towards efficient data valuation based on the shapley value," 2020.
[11] J. Weng, J. Weng, J. Zhang, M. Li, Y. Zhang, and W. Luo, "Deepchain: Auditable and privacy-preserving deep learning with blockchain-based incentive," *IEEE Transactions on Dependable and Secure Computing*, pp. 1–1, 2019.
[12] M. Shayan, C. Fung, I. Beschastnikh, and C. J. Yoon, "Biscotti: A ledger for private and secure peer-to-peer machine learning," *arXiv preprint arXiv:1811.09904*, 2018.
[13] X. Bao, C. Su, Y. Xiong, W. Huang, and Y. Hu, "Flchain: A blockchain for auditable federated learning with trust and incentive," in *2019 5th International Conference on Big Data Computing and Communications (BIGCOM)*, 2019, pp. 151–159.
[14] Y. Liu, Z. Ai, S. Sun, S. Zhang, Z. Liu, and H. Yu, "FedCoin: A Peer-to-Peer Payment System for Federated Learning," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2020.
[15] K. Toyoda, J. Zhao, A. N. S. Zhang, and P. T. Mathiopoulos, "Blockchain-enabled federated learning with mechanism design," *IEEE Access*, vol. 8, pp. 219 744–219 756, 2020.
[16] G. Radanovic, B. Faltings, and R. Jurca, "Incentives for effort in crowdsourcing using the peer truth serum," *ACM Trans. Intell. Syst. Technol.*, vol. 7, no. 4, Mar. 2016.
[17] Y. LeCun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 2, 1990, pp. 598–605.
[18] M. Courbariaux, Y. Bengio, and J. David, "BinaryConnect: Training deep neural networks with binary weights during propagations," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 28, 2015, pp. 3123–3131.

[19] F. Li, B. Zhang, and B. Liu, "Ternary weight networks," *arXiv preprint arXiv:1605.04711*, 2016.

[20] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[21] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Sparse binary compression: Towards distributed deep learning with minimal communication," in *International Joint Conference on Neural Networks (IJCNN)*, 2019, pp. 1–8.

[22] J. Xu, W. Du, R. Cheng, W. He, and Y. Jin, "Ternary compression for communication-efficient federated learning," *arXiv preprint arXiv:2003.03564*, 2020.

[23] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-iid data," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 9, pp. 772–785, 2020.

[24] D. Neumann, F. Sattler, H. Kirchhoffer, S. Wiedemann, K. Müller, H. Schwarz, T. Wiegand, D. Marpe, and W. Samek, "DeepCABAC: Plug & play compression of neural network weights and weight updates," in *2020 IEEE International Conference on Image Processing (ICIP)*, 2020, pp. 21–25.

[25] S. Wiedemann, H. Kirchhoffer, S. Matlage, P. Haase, A. Marbán, T. Marinc, D. Neumann, T. Nguyen, H. Schwarz, T. Wiegand, D. Marpe, and W. Samek, "DeepCABAC: A universal compression algorithm for deep neural networks," *IEEE J. Sel. Top. Signal Process.*, vol. 14, no. 4, pp. 700–714, 2020.

[26] F. Sattler, A. Marban, R. Rischke, and W. Samek, "CFD: Communication-efficient federated distillation via soft-label quantization and delta coding," *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 4, pp. 2025–2038, 2022.

[27] G. Hinton, O. Vinyals, and J. Dean, "Distilling the Knowledge in a Neural Network," *arXiv preprint arXiv:1503.02531*, 2015.

[28] E. Jeong, S. Oh, H. Kim, J. Park, M. Bennis, and S. Kim, "Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data," *arXiv preprint arXiv:1811.11479*, 2018.

[29] T. Lin, L. Kong, S. U. Stich, and M. Jaggi, "Ensemble Distillation for Robust Model Fusion in Federated Learning," *arXiv*, no. NeurIPS, 2020.

[30] S. Itahara, T. Nishio, Y. Koda, M. Morikura, and K. Yamamoto, "Distillation-based semi-supervised federated learning for communication-efficient collaborative training with non-IID private data," *arXiv*, pp. 1–11, 2020.

[31] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Consulted*, vol. 1, p. 2012, 2008.

[32] S. S. Shetty, C. A. Kamhoua, and L. L. Njilla, *Distributed Consensus Protocols and Algorithms*, 2019, pp. 25–50.

[33] V. Buterin, "Ethereum: A next-generation smart contract and decentralized application platform," 2013.

[34] K. Sekniqi, D. Laine, S. Buttolph, and E. GuenSirer, "Avalanche platform."

[35] J. Kwon and E. Buchman, "Cosmos whitepaper."

[36] Y. Zhao, J. Zhao, L. Jiang, R. Tan, D. Niyato, Z. Li, L. Lyu, and Y. Liu, "Privacy-preserving blockchain-based federated learning for IoT devices," *IEEE IoT Journal*, vol. 8, no. 3, pp. 1817–1829, 2021.

[37] Q. Zhang, Q. Ding, J. Zhu, and D. Li, "Blockchain empowered reliable federated learning by worker selection: A trustworthy reputation evaluation method," in *Proc. of WCNCW*, 2021, pp. 1–6.

[38] J. Kang, Z. Xiong, D. Niyato, S. Xie, and J. Zhang, "Incentive mechanism for reliable federated learning: A joint optimization approach to combining reputation and contract theory," *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 10 700–10 714, 2019.

[39] Z. Li, J. Liu, J. Hao, H. Wang, and M. Xian, "CrowdSFL: A secure crowd computing framework based on blockchain and federated learning," *Electronics*, vol. 9, no. 5, 2020.

[40] Fadaeddini, Amin and Majidi, Babak and Eshghi, Mohammad, "Privacy preserved decentralized deep learning: A blockchain based solution for secure ai-driven enterprise," in *Proc. of High-Performance Computing and Big Data Analysis*, 2019, pp. 32–40.

[41] C. He, B. Xiao, X. Chen, Q. Xu, and J. Lin, "Federated learning intellectual capital platform," *Personal and Ubiquitous Computing*, 2021.

[42] Y. Zou, F. Shen, F. Yan, J. Lin, and Y. Qiu, "Reputation-based regional federated learning for knowledge trading in blockchain-enhanced IoV," in *Proc. of IEEE WCNC*, Mar. 2021, pp. 1–6.

[43] K. Toyoda and A. N. Zhang, "Mechanism design for an incentive-aware blockchain-enabled federated learning platform," in *Proc. of International Conference on Big Data*. IEEE, 2019, pp. 395–403.

[44] S. Jiang and J. Wu, "A reward response game in the blockchain-powered federated learning system," *International Journal of Parallel, Emergent and Distributed Systems*, vol. 37, no. 1, pp. 68–90, 2022.

[45] J. Weng, J. Weng, J. Zhang, M. Li, Y. Zhang, and W. Luo, "DeepChain: Auditable and Privacy-Preserving Deep Learning with Blockchain-Based Incentive," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, pp. 2438–2455, 2021.

[46] L. Feng, Z. Yang, S. Guo, X. Qiu, W. Li, and P. Yu, "Two-layered blockchain architecture for federated learning over mobile edge network," *IEEE Network*, pp. 1–14, 2021.

[47] S. Xuan, M. Jin, X. Li, Z. Yao, W. Yang, and D. Man, "DAM-SE: A blockchain-based optimized solution for the counterattacks in the internet of federated learning systems," *Security and Communication Networks*, vol. 2021, p. 9965157, Jul. 2021.

[48] R. Zhang, M. Song, T. Li, Z. Yu, Y. Dai, X. Liu, and G. Wang, "Democratic learning: hardware/software co-design for lightweight blockchain-secured on-device machine learning," *International Journal of High Performance Systems Architecture*, vol. 118, p. 102205, Sep. 2021.

[49] 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, Aug. 2021.

[50] Y. Wang, Z. Su, N. Zhang, and A. Benslimane, "Learning in the air: Secure federated learning for UAV-assisted crowdsensing," *IEEE Trans. Network Sci.Eng.*, vol. 8, no. 2, pp. 1055–1069, Apr. 2021.

[51] Q. Hu, Z. Wang, M. Xu, and X. Cheng, "Blockchain and federated edge learning for privacy-preserving mobile crowdsensing," *IEEE Internet of Things Journal*, pp. 1–1, 2021.

[52] S. Ma, Y. Cao, and L. Xiong, "Transparent contribution evaluation for secure federated learning on blockchain," in *Proc. of ICDEW*, 2021, pp. 88–91.

[53] Y. Li, J. Zhang, J. Zhu, and W. Li, "Hbmd-fl: Heterogeneous federated learning algorithm based on blockchain and model distillation," in *Emerging Information Security and Applications*, J. Chen, D. He, and R. Lu, Eds. Cham: Springer Nature Switzerland, 2022, pp. 145–159.

[54] Y. Li, X. Tao, X. Zhang, J. Liu, and J. Xu, "Privacy-preserved federated learning for autonomous driving," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–12, 2021.

[55] J. Kang, Z. Xiong, D. Niyato, S. Xie, and J. Zhang, "Incentive mechanism for reliable federated learning: A joint optimization approach to combining reputation and contract theory," *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 10 700–10 714, 2019.

[56] M. H. ur Rehman, K. Salah, E. Damiani, and D. Svetinovic, "Towards blockchain-based reputation-aware federated learning," in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2020, pp. 183–188.

[57] A. Dasgupta and A. Ghosh, "Crowdsourced judgement elicitation with endogenous proficiency," in *Proceedings of the 22nd International Conference on World Wide Web*, ser. WWW '13, 2013, p. 319–330.

[58] B. F. Radu Jurca, "Incentives for answering hypothetical questions," 01 2011.

[59] B. Faltings, J. J. Li, and R. Jurca, "Incentive mechanisms for community sensing," *IEEE Transactions on Computers*, vol. 63, no. 1, pp. 115–128, 2014.

[60] J. Wang, C. Jiang, H. Zhang, Y. Ren, K.-C. Chen, and L. Hanzo, "Thirty years of machine learning: The road to pareto-optimal wireless networks," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 3, p. 1472–1514, 2020.

[61] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive federated learning in resource constrained edge computing systems," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, p. 1205–1221, 2019.

[62] Z. Liu, Y. Chen, H. Yu, Y. Liu, and L. Cui, "Gtg-shapley: Efficient and accurate participant contribution evaluation in federated learning," *ACM Trans. Intell. Syst. Technol.*, vol. 13, no. 4, may 2022. [Online]. Available: https://doi.org/10.1145/3501811

[63] A. Ghorbani and J. Zou, "Data shapley: Equitable valuation of data for machine learning," in *Proc. of the International Conference on Machine Learning (ICML)*, 2019.

[64] R. Jia, D. Dao, B. Wang, F. A. Hubis, N. Hynes, N. M. Gürel, B. Li, C. Zhang, D. Song, and C. J. Spanos, "Towards efficient data valuation based on the shapley value," in *Proc. of International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2019.

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

[66] F. Sattler, T. Korjakow, R. Rischke, and W. Samek, "FedAUX: Leveraging unlabeled auxiliary data in federated learning," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[67] G. Cohen, S. Afshar, J. Tapson, and A. van Schaik, "Emnist: an extension of mnist to handwritten letters," 2017.

[68] L. Deng, "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[69] A. Krizhevsky, "Learning multiple layers of features from tiny images," pp. 32–33, 2009. [Online]. Available: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf

[70] A. Coates, A. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, G. Gordon, D. Dunson, and M. Dudík, Eds., vol. 15. Fort Lauderdale, FL, USA: PMLR, 11–13 Apr 2011, pp. 215–223. [Online]. Available: https://proceedings.mlr.press/v15/coates11a.html

[71] H. Xiao, K. Rasul, and R. Vollgraf. (2017) Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms.

[72] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[73] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[74] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.

Leon Witt is a Ph.D. student at the Department of Computer Science and Technology at Tsinghua University in Beijing. He obtained a Master's degree in Mechanical Engineering and Business Adminstration from RWTH Aachen, Germany, with exchange semesters in Zurich and Los Angeles. He obtained a second Master's degree in Industrial Engineering at Tsinghua University in 2017. His research interests lie at the intersection of Federated Artificial Intelligence, Blockchain and Mechanism Design.

Usama Zafar is a Ph.D. student at the Department of Information Technology, Uppsala University, Sweden. He received a Master's (M.Sc.) degree in Computer Science from Tsinghua University in 2019 and a Bachelors of Engineering (B.E.) degree in Software Engineering from National University of Sciences and Technology (NUST) in 2015. His research interests include Distributed Machine Learning, as well as Security and Privacy issues in Federated Machine Learning.

KuoYeh Shen is a Ph.D. student at the Department of Computer Science and Technology at Tsinghua University in Beijing. He obtained a Master's degree of Mechanical Engineering and MBA of Finance at National Taiwan University. His research interests lie at the intersection of Federated Artificial Intelligence, Blockchain and Mechanism Design.

Felix Sattler received a M.Sc. degree in Computer Science in 2018, a M.Sc. degree in Applied Mathematics in 2018 and a B.Sc. degree in Mathematics in 2016, all from Technische Universität Berlin. He received his Ph.D. degree with distinction from the Technical University of Berlin in 2021. His research interests include Efficient and Robust Machine Learning, Federated Learning and Multi-Task Learning.

Dan Li is currently a full professor at the Department of Computer Science and Technology at Tsinghua University. He leads the NASP (Network Architecture, System and Protocols) research group, which is part of the networking research lab. He joined the faculty of Tsinghua University in March 2010, after two years working in the Wireless & Networking Group of Microsoft Research Asia as an associate researcher. His main research direction includes Trustworthy Internet, Data Center Network and Data-Driven Networking.

Songtao Wang is currently an assistant researcher at Zhongguancun Laboratory in Beijing, P.R.China. He received his Ph.D. degree of Computer Science and Technology from Tsinghua University in 2020 and Master degree of System on Chip from Southampton University in 2011. His research interests include Distributed Machine Learning, Data Center Network and Data-Driven Networking.

Wojciech Samek (M'13) is a professor at the Technical University of Berlin and Head of the Department of Artificial Intelligence at Fraunhofer Heinrich Hertz Institute (HHI), Berlin, Germany. He received a Diploma degree in Computer Science from Humboldt University Berlin in 2010, and his Ph.D. degree from the Technical University of Berlin in 2014. He is Fellow at the Berlin Institute for the Foundation of Learning and Data (BIFOLD), Senior Editor of IEEE TNNLS, and an elected member of the IEEE MLSP Technical Committee. He has been serving as an AC for NeurIPS'23, was a recipient of multiple best paper awards, including the 2020 Pattern Recognition Best Paper Award and the 2022 Digital Signal Processing Best Paper Prize. His research interests include Deep learning, Explainable and Trustworthy AI, and Federated Learning.