Cluster Computing (2024) 27:3997–4014
https://doi.org/10.1007/s10586-024-04273-1

A privacy-preserving federated learning framework for blockchain networks

Youssif Abuzied[1] · Mohamed Ghanem[1] · Fadi Dawoud[1] · Habiba Gamal[1] · Eslam Soliman[1] · Hossam Sharara[1] · Tamer ElBatt[1,2]

Received: 1 November 2023 / Revised: 3 January 2024 / Accepted: 4 January 2024 / Published online: 2 March 2024

Abstract
In this paper we introduce a scalable, privacy-preserving, federated learning framework, coined FLoBC, based on the concept of distributed ledgers underlying blockchains. This is motivated by the rapid growth of data worldwide, especially decentralized data which calls for scalable, decenteralized machine learning models which is capable of preserving the privacy of the data of the participating users. Towards this objective, we first motivate and define the problem scope. We then introduce the proposed FLoBC system architecture hinging on a number of key pillars, namely parallelism, decentralization and node update synchronization. In particular, we examine a number of known node update synchronization policies and examine their performance merits and design trade-offs. Finally, we compare the proposed federated learning system to a centralized learning system baseline to demonstrate its performance merits. Our main finding in this paper is that our proposed decentralized learning framework was able to achieve comparable performance to a classic centralized learning system, while distributing the model training process across multiple nodes without sharing their actual data. This provides a scalable, privacy-preserving solution for training a variety of large machine learning models.

✉ Youssif Abuzied
youssif.abuzied@aucegypt.edu

Mohamed Ghanem
oscar@aucegypt.edu

Fadi Dawoud
fadiadel@aucegypt.edu

Habiba Gamal
habibabassem@aucegypt.edu

Eslam Soliman
eslam98@aucegypt.edu

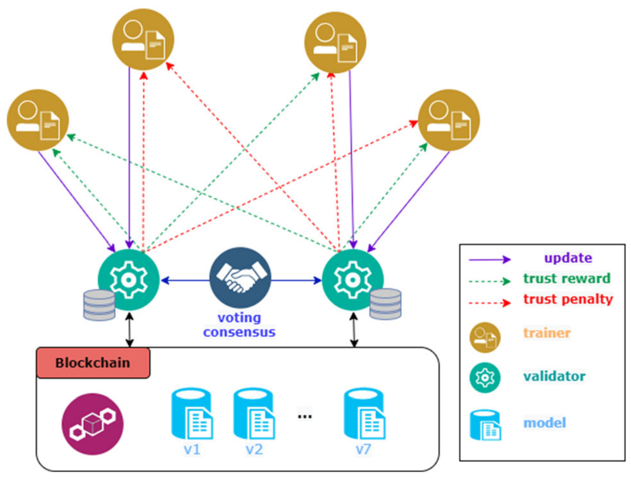
Hossam Sharara
hossam.sharara@aucegypt.edu

Tamer ElBatt
tamer.elbatt@aucegypt.edu

1 Computer Science and Engineering Department, The American University in Cairo, Cairo, Egypt

2 Electronics and Communications Engineering Department, Cairo University, Giza, Egypt

Graphical abstract



Keywords Federated learning · Blockchains · Decenrtalized data · Data privacy

1 Introduction

Recently, there has been a growing interest, in the computing and machine learning research communities, in scaling and speeding up machine learning processes, especially the training part. This is motivated by the rapid growth of data, especially decentralized data, which cannot be stored at a single data center either due to its size or its nature of being generated and stored at dispersed entities.

Centralized machine learning has witnessed major success over the past decade thanks to the wealth of collected data sets along with the major developments in processing technologies attributed primarily to breakthroughs in the design and optimization of Graphical Processing Units (GPUs) and memory technologies. Nevertheless, the unmatched, growing demand for machine learning applications calls for scalable and privacy-preserving learning paradigms departing from classic centralized learning models which are trained and tested on a single machine where all data is stored.

Federated learning (FL) is a machine learning paradigm conceived at Google in 2016 [1]. It proposes a semi-distributed learning framework where individual machines train local models on decentralized data and then share their model weights/parameters with the FL server which, in turn, combines (or aggregates), e.g., through averaging, the model weights and feeds them back to the individual machines to refine their models for enhanced prediction accuracy. FL is well-known for its privacy-preserving merits (since data is processed locally, not at the server) and also for its scalability merits (especially for big, decentralized data where storage and processing at a single data center becomes prohibitive).

However, Federated learning systems face a lot of challenges. First, communication cost is a key bottleneck in federated learning systems as local nodes have to send model updates for aggregation and verification. Second, a key security threat that federated learning systems face is having a single point of failure. Third, federated learning systems should accommodate scenarios where some local trainers produce low quality updates for different reasons such as low quality local training data. Moreover, federated learning systems should have a methodology for dealing with trainers sending poisonous updates [2].

Our contribution to address the above challenges is multi-fold. First, design and architect a novel framework for federated learning-based systems on top of blockchain technology, coined FLoBC [3]. This architecture enables our proposed framework to involve the data owners in the model development without jeopardizing the privacy of their data. Hence, it fast tracks the training and development of machine learning algorithms with a much larger scale of data that was hard to collect before. Second, we examine the proposed framework with two iterative optimization techniques, namely classic stochastic gradient descent (SGD) adopted widely in the machine learning literature and Bayesian Optimization (BO). The choice of these two algorithms is motivated by the need to cover a variety of machine learning algorithms that rely on these techniques in their optimizations. Finally, we develop an experimental setup trained on a public data sets to show the performance merits and design trade-offs associated with our proposed system. In this work, we further illustrate how the proposed framework offers solutions to common challenges in decentralized federated learning, namely, **privacy**, **efficiency**, and **Byzantine fault-tolerance**. FLoBC is a proof-of-concept that spawns mini-systems to demonstrate diverse aspects of our framework at a small scale. Later in the paper, we conduct several experiments in order to verify its merits compared to a baseline centralized machine learning system.

The remaining of this paper is organized as follows. In Sect. 2, we give the present the background and literature survey. Afterwards, we introduce the methodology and proposed federated architecture in Sects. 3 and 4. In Sect. 5, we present the experimental setup, performance results and discussion. Finally, in Sect. 6, we draw the conclusions and point out potential directions for future research.

2 Background and related work

Prior to the introduction of *federated learning* (FL) in 2016, the framework of *blockchains* has been a prominent technology that aimed at realizing decentralized consensus [4]. It is envisioned that combining the blockchain and federated learning technologies holds a great promise to achieve more decentralization and privacy. A number of contributions in the area of federated learning involve blockchains while others employ different decentralized protocols [5–7]. The key objective is to eliminate the need for a centralized server that collects the data and performs model training. The motivation for that is two-fold. At one hand, it guarantees privacy due to processing the data on local machines where it is originally stored, and on the other hand, it eases the minimum computation requirements by distributing the extensive machine learning computation burden across the network. Part of the prior work rely on an All-Reduce scheme in which each training node shares its updates with all the nodes in the system resulting in a communication cost of $O(n^2)$, for n training nodes. One important aspect to leverage, when applicable, is the network topology which can be exploited to reduce the communication cost. For instance, prior proposals in the literature adopt a variety of topologies for a node to share its model updates with its one-hop neighbors, e.g., ring topology in [8], tree topology in [9] and graph topology in [10, 11]. However, all these approaches require multi-hops for the updates to reach all the nodes in the network, thus leading to slow convergence.

Motivated by the same objective of reducing the communication overhead, [12] utilizes a decentralized communication protocol and personalized sparse masks to tailor local models. This approach effectively reduces the communication overhead by maintaining a fixed number of active parameters during training and peer-to-peer communication. Moreover, [13] adopts the gossip protocol whereby a node sends its model updates to a peer node in the network, thus propagating the updates across the entire network. Another approach adopted by [14] is encrypting the updates and the global model prior to sending them to all the other peer nodes which in turn de-crypt the updates and the global model. The common factor across all these approaches is that they rely on the trainer to validate the updates sent by peer nodes, which puts more complexity on the trainers, as well as introduce security vulnerabilities in the system in case of malicious actors. To resolve these issues, our proposed FLoBC framework introduces "validators" (similar to cryptocurrency miners) in order to offload the computational burden of the trainers and reinforcing the security of the decentralized system.

The use of blockchains as the underlying networking infrastructure for decentralized federated learning systems has also been introduced in the literature, with the objective of utilizing the blockchain inherent features for ensuring the model updates are immutable and secure, and facilitate the traceability of the updates through proof-of-work architectures. In [15], the role of the central server in centralized federated learning is achieved by the smart contracts. An arbitrary node performs local training. Afterwards, the local model updates are sent to an elected consensus committee that verifies and assigns scores to the model updates. Subsequently, the updated global model is incorporated into the blockchain. In [16] and [17], each training node is assigned a validator that validates the trainer's updates, computes proof of work (PoW), and rewards the trainer accordingly. [18] adopts a similar approach but the trainer sends its updates to an any validator and the rewards are granted by the model owner, not the validator or the blockchain. An alternate direction in [19] and [20] employs distributed peer-to-peer file sharing to store the model checkpoints. [17, 19, 20] are specially tailored for automotive in-vehicle training. In our proposed framework, we propose the employment of *proof of stake* (PoS) [21] as a lightweight alternative to cryptographic Proof-of-Work, to facilitate the communication and optimize the system performance. We also focus on providing a generic framework with a basic set of consensus assumptions that can be utilized for any machine learning task, rather than building a specialized architecture for a certain class of problems.

3 Blockchain-based federated learning framework

In this section, we will introduce the principles and methodology for our proposed blockchain-based federated learning framework. Next, we will discuss the three main pillars for our proposed model, namely parallel training, decentralization, and synchronization.

3.1 Parallel training

The first challenge is how to realize parallel model training to achieve the desired performance, while maintaining the target level of generality which allows the proposed framework to be compatible with any machine learning algorithm relying on gradient descent or bayesian optimization.

The two main paradigms for parallelizing machine learning models are: Model-Parallelism and Data-Parallelism [22]. Under Model-Parallelism, the model structure itself is split or distributed across multiple nodes, a paradigm often referred to in the literature as Split Learning [23]. While Split Learning has the advantages of preserving data privacy and enabling nodes with low computational power to participate in the training, it is not the best suited for our framework due to its requirement for training the data sequentially for each sub-model. This, in turn, ultimately limits the model scalability [24] and directly impacts the generality aspect we are targeting because of its dependence on the actual model being used for training.

On the other hand, under the Data-Parallelism paradigm, each node trains the entire model but only on a subset of the training data. This eliminates the dependency on a specific machine learning model being used, and the sequential limitation of Model-Parallelism. After each model is trained at each individual node, in parallel, the generated models' updates are aggregated into the global model. As described, the Data-Parallelism paradigm cab be readily generalized to any process that can be divided into sub-processes and then aggregating their outcomes. This suits processes such as gradient computation and probability scores calculations [25], which are at the core of many machine learning models.

Thus, the choice of the iterative optimization technique for our proposed federated learning framework has to factor in the property that it can be approximated through sub-computations at the distributed nodes. For instance, traditional gradient descent based optimization is not suitable for our, or any other, distributed framework as the gradients are computed on the entire data set throughout the training process. On the contrary, its stochastic variant (stochastic gradient descent) is more suitable due to its ability to approximate the true gradients through local gradients computed at each node. Another alternative would be Bayesian optimization which relies on probability computations of the entire data set, which, again, can be approximated through aggregating the probabilities at each node with the respective weights of the data available at the respective nodes. In order to use a generalizable method of aggregation, we follow the same approach of Federated learning systems through Federated Averaging of the local weights to come up with the global model weights. This process has been proven to converge to the correct global weights (being it the true gradients or the true probabilities) through the central limit theorem; different nodes training on different data sets whose updates get aggregated or averaged onto a single model would eventually converge to a valid global model.

3.2 Decentralization

While classic federated learning models utilize a decentralized set of computational nodes to carry out the training process, they still require the use of a central server or cluster of servers supervising the training done by other nodes. In our proposed model, we aspire to reach a higher level of decentralization that eliminates the need for a central server altogether.

Towards this goal, we resort to blockchain networks as a decentralized infrastructure with no central server co-ordinating the process. Thus, the individual nodes are trained in a fully decentralized manner. Similar to most cryptocurrency networks in the literature, we design our framework to have two types of nodes: **trainers** and **validators**. The **trainer** nodes are the ones who carry out the model training process, similar to the process carried out in classic server-based federated learning, hence, achieving the level of data privacy desired, where each node trains the model on its own data without sharing it with other nodes.

On the other hand, the **validator** nodes act as gate keepers to make sure that the trained models produced by the trainer nodes are of acceptable quality, thus preventing both malicious nodes and ones with low data / model quality from contaminating the aggregate, generalized model, generated collaboratively. This can be achieved by having the validator nodes assess the quality of the generated model through independent training data that is not shared with the trainers. This validation process is done collectively through decentralized consensus based on voting to decide whether a trainer update is acceptable, and should be used to update the generalized model, or not.

In order to guarantee that the desired process works as expected, the following requirements should be met:

- Strictly more than 2/3 of validators are non-Byzantine.
- Different validators' data sets should be sufficiently positively correlated in order to reach a meaningful consensus.
- The underlying network is at least partially synchronous.
- The quality of models produced by the system is limited by the amount of collective useful data used during training and validation.

3.3 Synchronization

The last challenge for our proposed blockchain-based federated learning framework is how often to synchronize the global model updates, referred to as the synchronization barrier. On one hand, if the model updates are happening slowly using low frequency synchronization barrier, this leads to model degradation as the trainers' parameters will be out of sync [22]. This, in turn, leads to slower convergence. On the other hand, if the model updates are happening too often using a high frequency synchronization barrier, the model will converge faster [23], but at the expense of a significantly higher communication overhead, due to the increased network and node bandwidth usage.

In order to strike a balance for this trade-off, there are multiple approaches in the literature to tackle this problem. In the rest of this section, we will introduce the most common techniques and discuss the advantages and limitations of each, and their suitability for our proposed framework.

Bulk Synchronous Parallel (BSP) The first technique, abbreviated as BSP, is known to be the simplest to preserve consistency by alternating between a sequence of rounds of computation followed by communication. The main advantage of this model is that it has the best consistency (hence, fastest convergence), yet, its main drawback is that the finished nodes will have to wait for all other worker nodes to finish their computation at each synchronization barrier [22]. In our variant of BSP, each training round is limited by a fixed period that ends with a strict deadline after which no further trainer updates are considered for that round.

Stale Synchronous Parallel (SSP) This model is similar to BSP but with more relaxed synchronization barriers that allow a certain number of iterations beyond the preset number. After that leeway is reached, all workers are paused [23]. Given its nature as a compromise to BSP, it has good convergence characteristics in low to moderate staleness, beyond which convergence rates start to decay [22]. In our proposed framework, SSP is modelled as BSP with the possibility of a deadline extension that is a fraction of the original round period determined in proportion to the ratio of trainers that have not yet finished training for that round, denoted by *slack ratio*. Meanwhile, trainers that have finished by the deadline are allowed to continue training for at most N more steps before the deadline extension expires.

Barrierless Asynchronous Parallel (BAP) This synchronization approach represents the opposite end of the spectrum to BSP as it almost totally eliminates the cost of synchronization by allowing asynchronous communication between nodes (with no wait), which achieves a very low communication overhead (hence, more efficient bandwidth usage). However, it exhibits potentially slow and even incorrect convergence with increased delays [22]. In our proposed variant of BAP, trainers are allowed to train for as long as a training round may extend; that is, until a new model is released. Trainers can keep advancing local training, periodically sharing it with a validator. A new model is released when a minimum ratio of labor has been submitted, and that's when trainers should pull in the new model.

Approximate Synchronous Parallel (ASP) Although our system does not support ASP, the scheme remains note-worthy. Unlike the SSP model, ASP is concerned with limiting parameter accuracy instead of staleness. It does so by ignoring updates that are not significant to spare the synchronization cost for them [22]. However, one downside of this approach is that it is not easy to decide which parameters are insignificant, along with increased complexity of implementation.

3.4 Byzantine fault-tolerance

Possible cases and scenarios of Byzantine behavior are countless, and decentralized systems are arguably most prone to it. In this work, we focus on lazy[1] and malicious trainer behavior given that a malicious validator behavior is mostly taken care of by the blockchain's decentralized consensus. To remedy this, we employ a reward-penalty policy, under which each trainer i is assigned a trust score (or a reputation) ϕ_i such that:

$$0 \leq \phi_i \leq 1$$

$$\sum_{i=1}^{n} \phi_i = 1$$

A trainer's trust score is used as the weight factor for that trainer updates in the decentralized federated learning algorithm. The profound effect of this is two-fold. First, it

[1] Lazy, in this context, includes nodes that are not contributing useful updates to models.

enables validators to control the impact of said trainer's updates on the model based on its trust level. Second, it creates intrinsic competition in the decentralized system since any rise in one trainer's trust score effectively results in a relative decline in other trainers' trust scores. Trust scores are adjusted based on the validation results. Upon receiving a trainer update, a validator adds its gradients or probability scores to the latest model weights, and computes its validation score. Subsequently, it updates the respective trainer's trust score based on whether it leads to improvement or decline.

4 FLoBC architecture

In this section, we will present an overview of the architecture of our proposed framework, including the various services the system provides, the underlying blockchain architecture, and a proof-of-concept implementation.

4.1 System overview

FLoBC is a distributed system driven by two main actors: trainers and validators, the latter of which has more involved responsibilities. On a lower level, the system is composed of six main services, each providing a very specific functionality and has a strictly defined interface with other services and layers.

1. **Blockchain service** Constitutes the main fabric of communication and data storage across all validator nodes. To ensure data consistency across all validators, this subsystem utilizes an elaborate schema for decentralized consensus, namely a variant of the Practical Byzantine Fault Tolerance (pBFT) family of consensus algorithms that are mainly based on voting and elections.
2. **Storage schema service** Facilitates and enables structured access and modification of the local storage database on which the blockchain state is stored.
3. **Machine learning service** Represents the machine learning layer on top of the blockchain. It is built as a service that handles the execution of machine learning transactions such as gradients or probability scores sharing by interfacing with the local storage schema to reflect the transaction execution such as model creation and model aggregation.
4. **Training validation service** Is a hybrid layer between the blockchain and the training layer. It is responsible for validating incoming updates and returning a verdict of whether a transaction should be accepted or rejected.
5. **Reputation management service** Handles the accounting aspects of the system, realizing a reward-punishment mechanism to ensure that only honest trainers get to stay in the network while Byzantine ones are dismissed to maintain the performance and quality of created models and ensure fairness.
6. **Model training service** Performs model training including all needed intermediate transformations such as model flattening and rebuilding at the trainer side. This layer sends and receives flattened models and flattened model gradients or probability scores.
7. **Training flow management service** Manages the communication between trainers and validators including model and gradients or probability scores exchanging, along with synchronizing training iterations.

4.2 System implementation

For our proof-of-concept framework implementation, we chose the Exonum Blockchain [26] as the underlying communication architecture. The rationale behind this choice is the fact that Exonum Blockchain already utilizes a variant of the practical Byzantine Fault Tolerance (pBFT) consensus algorithm, which is the algorithm of choice for our framework. In addition, the Exonum Blockchain provides a voting-based consensus algorithm that is lightweight compared to the typical cryptographic Proof-of-Work (PoW) consensus [26]. This lightweight algorithm provides a crucial advantage of our implementation to mitigate the heavy computing weight of the machine learning model training. For the remaining system services, we relied on the *Rust* programming language for implementation, and Javascript for the lightweight clients. Finally, for the training and validation of machine learning models, we provide an API for using any Python script to achieve our framework's generality and to ensure we can plug and test any machine learning model for our and future experiments.

4.3 System views

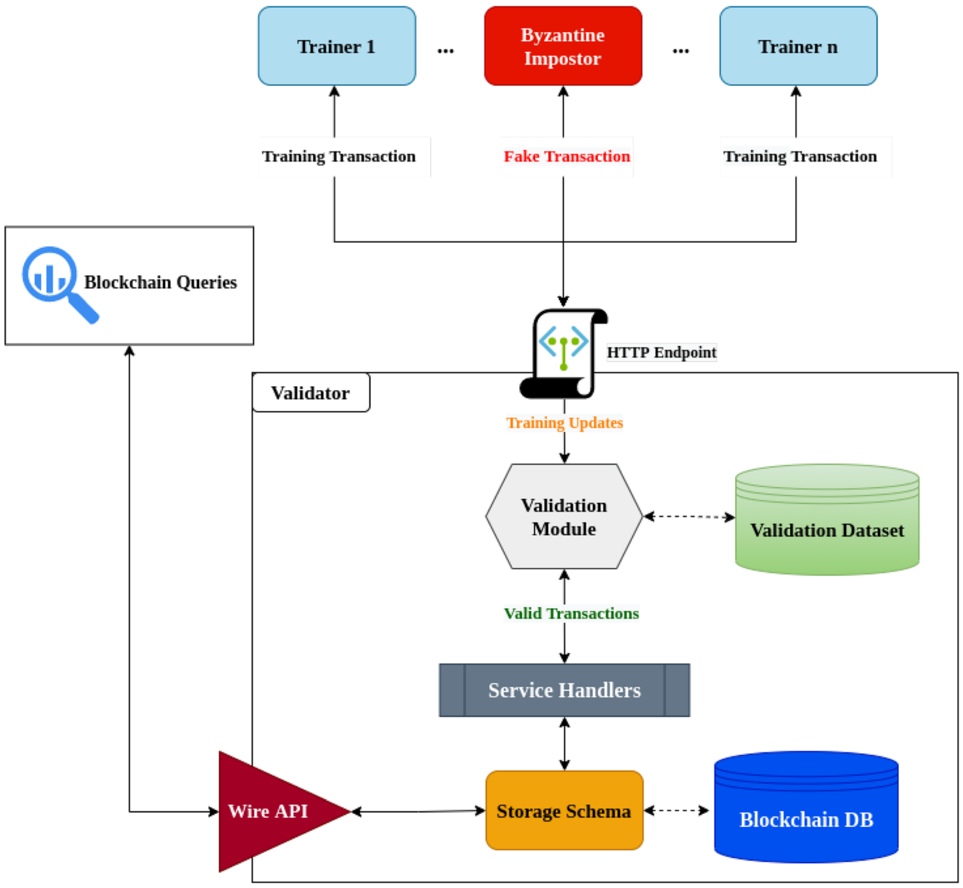
There are multiple ways to view FLoBC depending on the scope level, the most important of which are presented in the following.

4.3.1 Actor-level view

This view is the most high-level depiction of the system in terms of its main acting entities and how they are connected.

In our system, we have two types of actors: trainer and validator. While trainers constitute the main work labor in the system by performing gradient or probability scores calculations using their data, validators are even heavier in

Fig. 1 Modular view



composition as they are the ones undertaking elections to achieve consensus and validation to ensure model quality.

One important property of our architecture is that trainers are not fully connected to the validators while validators are fully interconnected. It is worth noting that while the roles (i.e., trainer or validator) are not fixed or dedicated, a node is not allowed to perform both roles simultaneously for the same subnetwork, yet a node can be a validator on a subnetwork while being a trainer for another. Further, each model subnetwork requires at least one validator to function.

4.3.2 Modular and service-level views

The modular view in Fig. 1 gives a somewhat closer look on a validator's insides, showing the data flow from each high-level module to the others. It first shows that trainers communicate with validators over HTTP whose endpoint then passes the transactions to the validation module that uses the validation dataset to validate the model-update transactions in order to ultimately decide whether to accept of reject the update onto the blockchain. Validators also expose a Wire API for answering queries about the blockchain state (e.g., what's the latest model version?), which is helpful for system diagnosis/monitoring. On the other hand, the service-level view emphasizes the separation of concerns/responsibilities across the system into multiple services illustrated in Fig. 2. The storage schema service is central to all services in the validator side as it represents the main interface for making persistent changes. The ML service is responsible for consolidating trainer updates and handling the trainer flow (e.g., synchronization) while the validation service is responsible for assessing trainer work and providing the results to the reputation service which can adjust trust scores accordingly. The Wire APIs on the sides represent auxiliary querying interfaces for miscellaneous purposes (e.g., for getting the latest released model version) besides the main blockchain interfaces used for sending transactions.

5 Experiments and performance results

In this section, we present the experiments carried out using the proposed framework. As our framework supports both gradient descent and bayesian optimizations, we decided to test both approaches through a set of experiments utilizing a Convolutional Neural Network (CNN) to

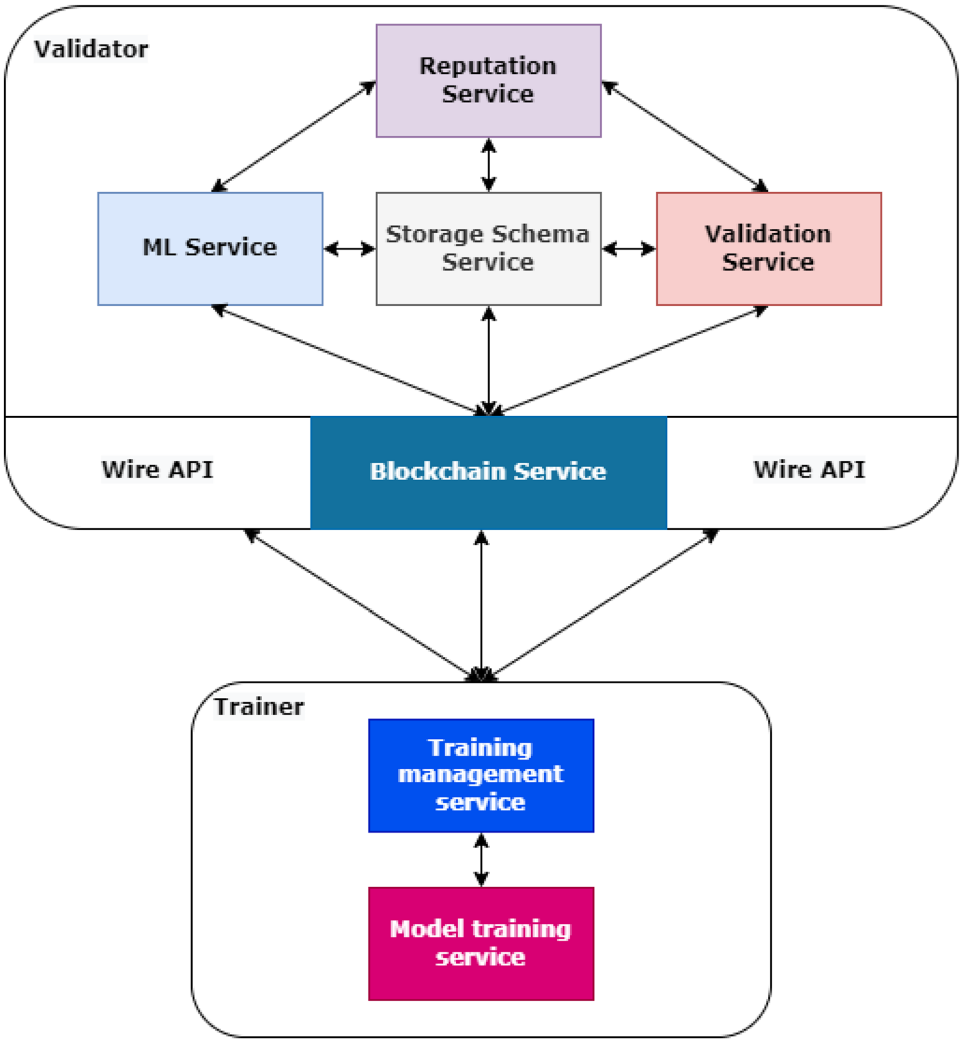


Fig. 2 Service-level view

validate the framework's performance on gradient descent and another set of experiments using a Bayesian Network to validate the performance of bayesian optimization. Moreover, we included another set of experiments to test our proposed method against other state of the art methods to ensure our method's efficiency. For the first set of experiments, we utilized the MNIST [27] data set to train a CNN for predicting handwritten digits. For the second set of experiments, we utilized the Alarm Network [28] data set adopted from the Bayesian Network Repository. Both data sets were split randomly across the trainers to achieve the data decentralization principle. Finally, for the last set of experiments, we replicated some experiments mentioned in the papers that introduced the state of art method for our method and test the performance we got against the performance reported in those papers. The metric used to evaluate training in these experiments is accuracy, but the system will work equally well using any machine learning model metric. The metric is easily pluggable into the Python machine learning validation script.

5.1 Benchmark: decentralized vs centralized performance

In this experiment, we compare the performance of the centralized model with that of the decentralized model utilizing the best performing configuration (as demonstrated later by experiment 1) comprising 7 trainers and 3 validators with an active reward-penalty policy. The centralized model is the golden benchmark of our system. The aim of this experiment is to test whether the added benefits of decentralization would result in a significant cost in terms of model quality.

5.1.1 Setup

In this experiment, the centralized model uses the whole data available in the MNIST training dataset. Since in the best performing decentralized configuration there are 7 trainers affecting the model each iteration, in our centralized benchmark, each training iteration is 7 epochs. The

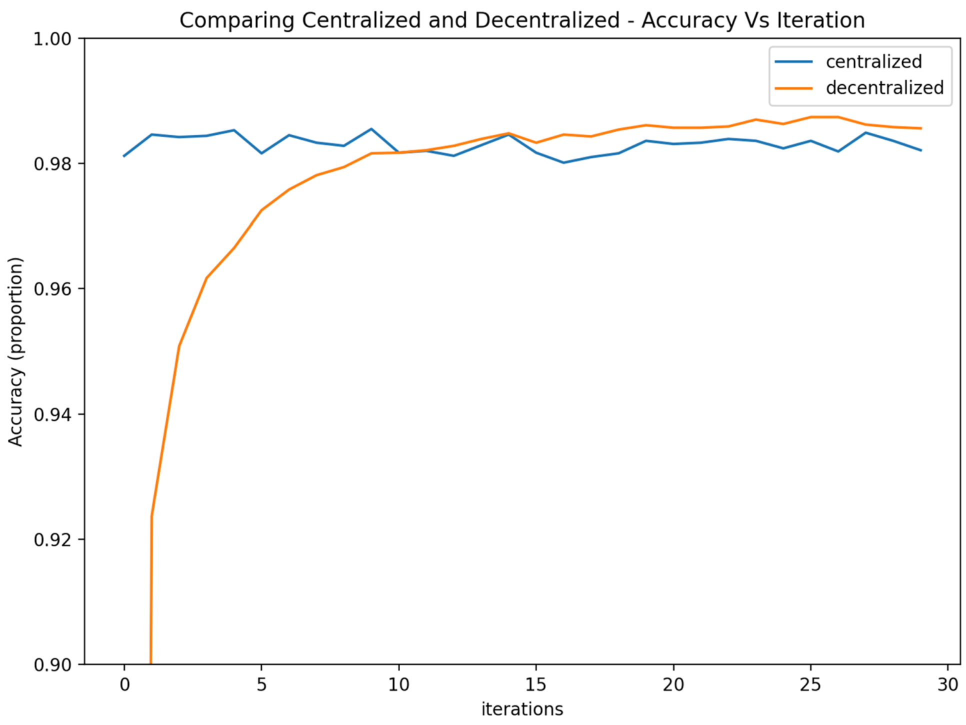


Fig. 3 Accuracy against iterations for the centralized and decentralized runs

training lasts for 30 iterations for both the centralized and decentralized runs.

5.1.2 Results and discussion

As it can be seen in Fig. 3, the centralized model had a higher accuracy than the decentralized alternative for 10 iterations, after which the decentralized model trained by 7 trainers and validated by 3 validators marginally outperformed the centralized model. The difference in accuracy is below 0.5%. This experiment shows that the added benefits of decentralization and privacy preserving training do not degrade the federated model performance. Note that this is not meant to demonstrate that the decentralized version is superior in performance to its centralized counterpart. Rather, it shows that the two are closely comparable in that regard.

5.2 Experiment 1: Trainers-to-validators ratio

In this experiment, we have a total of N participating nodes. In each run, we split N into different numbers of trainers and validators. Through this experiment, we wish to determine the best trainers-to-validators split that makes use of the most computing recources and achieves the best model performance while still relying on multiple validators to generate a more trustworthy model. This experiment will also show the speed of convergence to the highest accuracy by pointing out the iteration at which the highest accuracy was achieved. There is a trade-off between the number of trainers and validators that this experiment aims to balance out. Note that increasing the number of trainers with good quality data should increase the model performance. Greedily, we would want all the nodes in the system to be trainers, however, this has a major drawback. Relying on a single validator means that the validator is a trusted entity whereas relying on multiple validators increases the trustworthiness of the model because of consensus. However, we want to refrain from having too many validators because this wastes computing resources that could have been utilized in training and adds on unnecessary communication costs. A trainer shares its updates with one validator that validates those updates then shares them with the rest of the validators for further validation and finally consensus. This being said, as we increase the number of the validators, we increase the communication cost associated with voting consensus. For these reasons, for the highest efficiency and the best performance, it is important to determine the best trainers-to-validators ratio that achieves the best model quality without compromising decentralization.

5.2.1 Setup

In this experiment, the N number of agents in the system is set to 10. For instance, with N = 10, one system run has 9 trainers and 1 validator while another run has 8 trainers and 2 validators, etc. Each run, we change the number of trainers and validators to try all their combinations that add

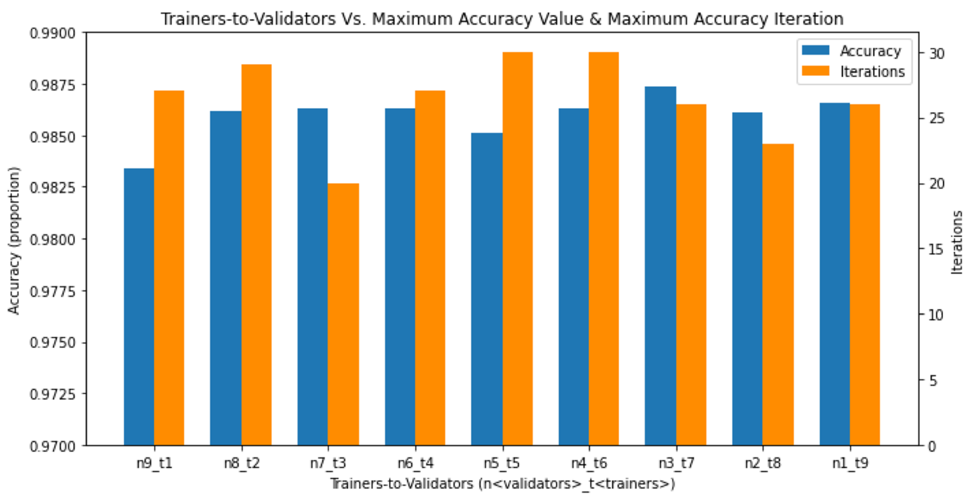


Fig. 4 Maximum accuracy and its iteration index for different splits of trainers and validators

up to 10 agents. Each run lasts for 30 iterations. We record the iteration and its accuracy for each run of the system. The new model version creation waits for all the trainers to share their updates with the validators since it is important in this experiment that all trainers submit their work for validation each model update iteration since we are studying the effect of the split of the validators and trainers on the model performance.

5.2.2 Results and discussion

As it can be seen from Fig. 4, the best configuration for this model is having 3 validators and 7 trainers. Using this configuration, the maximum accuracy of 0.9874 was reached at iteration 26. Note that the least accuracy was that for the single trainer configuration since much less computing resources and data have been dedicated for training.

5.3 Experiment 2: Reward-penalty policy

The purpose of this experiment is to evaluate the usefulness of computing a score for each trainer, where a trainer's score affects the extent to which this trainer's updates affect the overall model.

5.3.1 Setup

This experiment is performed using processes representing 6 trainers (indexed from 0 to 5) and 3 validators. Training ran for 30 iterations. The BSP synchronization scheme was employed with a sufficiently large period to allow all trainers to submit updates every synchronization period.

The updates of each of the trainers were offset using an approximately normal noise. The mean of the noise is 0 and the standard deviation of the noise is linearly proportional to the index of the trainer, with proportionality constant k = 0.0545. Thus, the updates of trainer 0 received zero noise, and the updates of trainer 5 were significantly offset by the noise.

For purposes of this experiment, the minimum acceptance threshold for updates was reduced substantially to allow low-performing updates to affect the model generation and be affected by scoring. The control group had scoring disabled; that is, the scores of all 6 trainers were constant and equal throughout the training process. The scoring group allowed scoring for each trainer to change and to adapt to the relative qualities of the provided updates by each trainer.

Table 1 Scores of all trainers across training rounds

Round	0	5	10	15	20	25	30
Trainer 0	0.16667	0.764622	0.823812	0.866141	0.907603	0.964613	1
Trainer 1	0.16667	0.235378	0.176188	0.133859	0.092397	0.035387	0
Trainer 2	0.16667	0	0	0	0	0	0
Trainer 3	0.16667	0	0	0	0	0	0
Trainer 4	0.16667	0	0	0	0	0	0
Trainer 5	0.16667	0	0	0	0	0	0

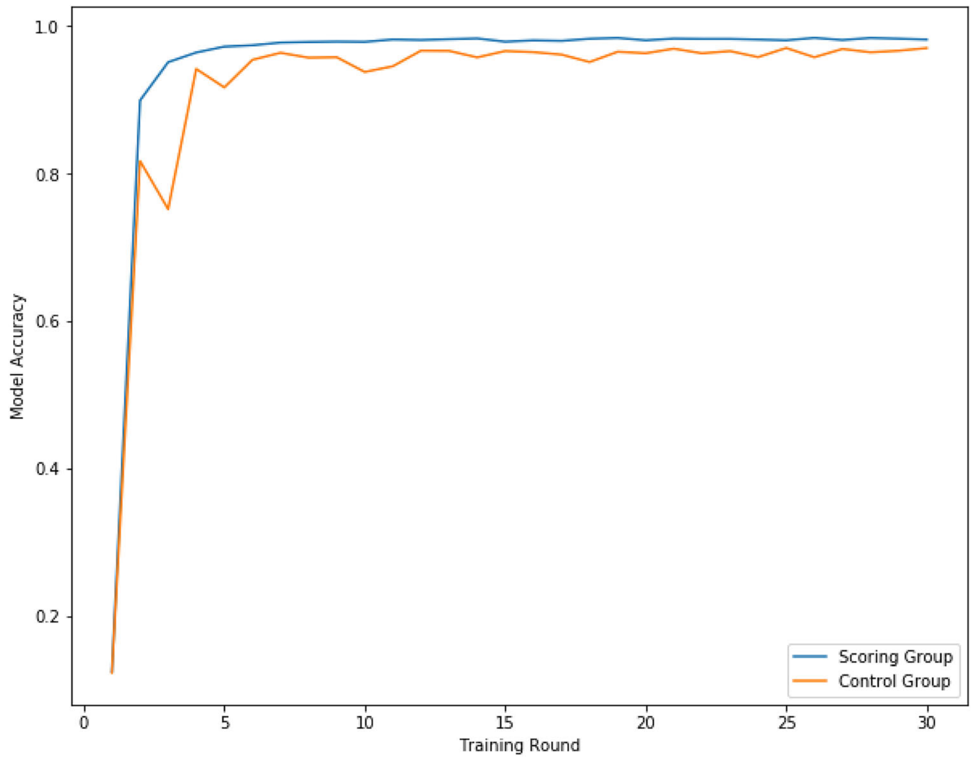


Fig. 5 Accuracy performance of scoring group vs control (uniform) group over 30 training rounds

5.3.2 Results and discussion

During the operation of the scoring group, the scores of each trainer were updated after every synchronization period. A sample of the scores of all trainers over the 30 training iterations is displayed in Table 1. We note that, shortly after training started, the validators realized that most trainers were providing low-quality updates, and thus, the scores of four trainers dropped to zero almost immediately. Over the remaining rounds, the score of trainer 0, whose updates were not affected by added noise, steadily increased against the score of the second-best trainer, trainer 1, whose updates were affected by a minimal amount of noise. Hence, we conclude that scoring enables the validators to accurately estimate the relative performance of each of the trainers.

For the two groups, after each training round, the current model accuracy is computed. The accuracy performance of the two groups is shown in Fig. 5. We note that trainer scoring improved the performance, stability, and convergence speed of the model, as only the updates from high-performing trainers were taken into consideration and contributed to improving the model.

5.4 Experiment 3: Synchronization schemes

In this experiment, we compare some of the common synchronization schemes used in parallel computing after adapting our own variations of them. The schemes implemented in FLoBC and those tested in the experiment are Bulk Synchronous Parallel (BSP), Stale Synchronous Parallel (SSP), and Barrierless Asynchronous Parallel (BAP), as previously described in the methodology. The purpose of the experiment is to compare the different schemes in terms of two main metrics: model growth relative to the number of training rounds, and the average time per training round. The former indicates convergence while the latter indicates progression speed.

5.4.1 Setup

For the experiment setup, a system of three validators and six trainers is used in all runs with a uniform trust policy; that is, all trainers have the same trust scores. For both BSP and SSP, the same base synchronization barrier period is used. Different trainers are running at different paces to emulate real-world computational differences, with the synchronization barrier period set to be longer than the time most (but not all) trainers will take to finish one training job. To measure the model growth metric, the system will run for a fixed N=30 training rounds, comparing how long each scheme takes to converge on the highest model accuracy while also taking the three highest accuracies into consideration. As for the average training iteration delay, the system will run for a fixed period of 20 min. For BAP, we use two configurations: one with a

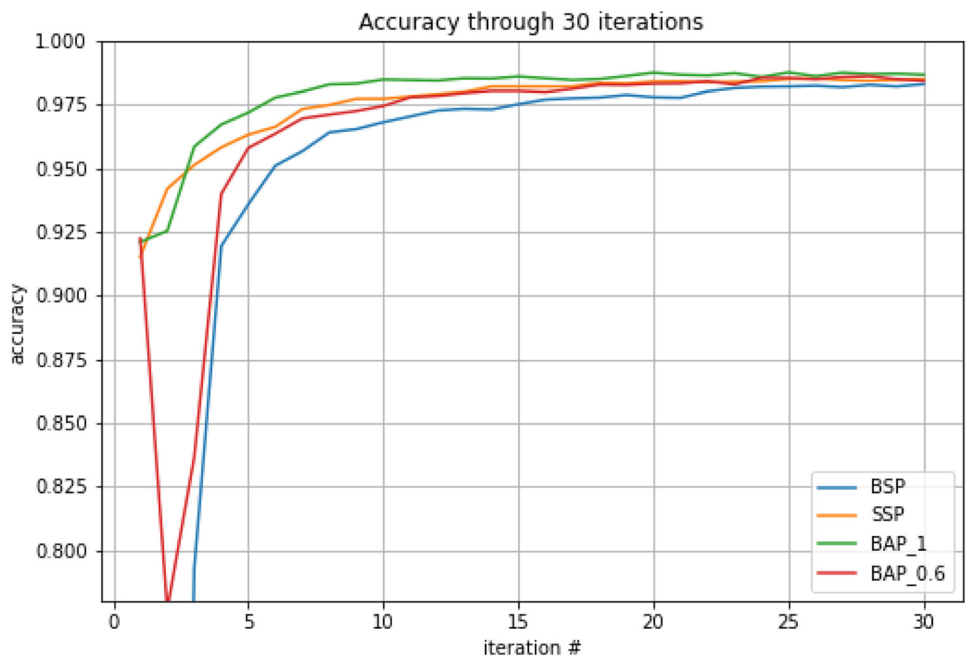


Fig. 6 Accuracy performance for 4 different schemes across 30 training iterations

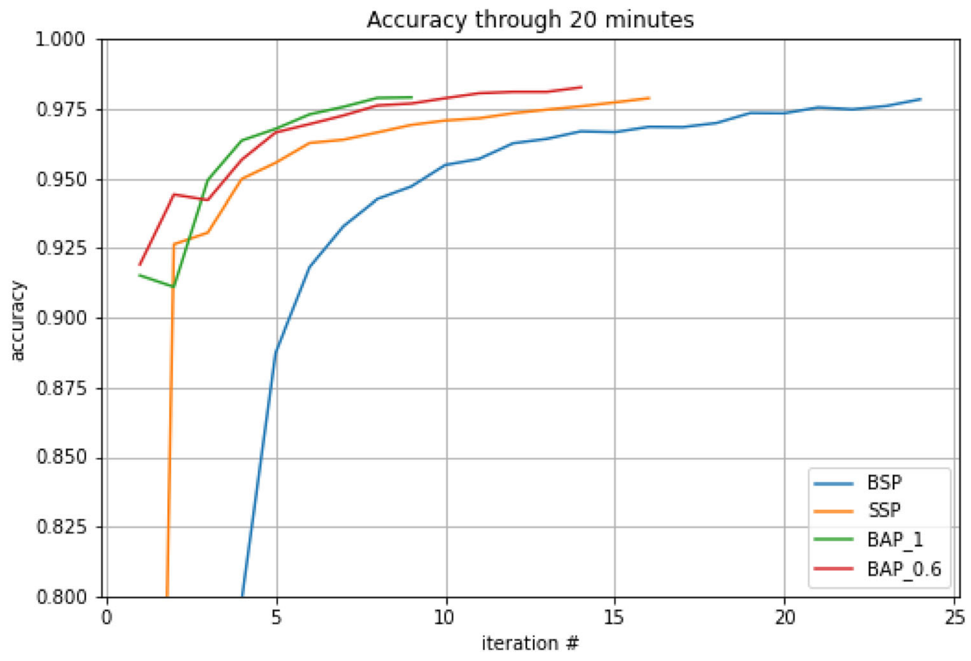


Fig. 7 Accuracy performance for 4 different schemes across 20 min of training time

slack ratio threshold of 0% (fully relaxed), and another with 40%. That is, in the first one, a sync barrier is thrown when all trainers have submitted at least one update in the current training round while the second one is contingent on only 60% of trainers having done so.

5.4.2 Results and discussion

As depicted in Fig. 6, results seemingly meet their theoretical expectations, confirming the trade-offs for the three different schemes. BSP has the lowest performance as it is the most strict in terms of synchronization, leading to lower utilization of training power. However, it is quite stable as it offers more predictable limits on speed of progression (at a fixed period). As for the two BAP runs,[2] BAP_1 with a majority ratio of 100% was expected to perform best since it has the best utilization of trainer work, beating its BAP_0.6 counterpart, which confirms the expected effect of decreasing the majority ratio in our variant of BAP. On

[2] Labelled as BAP_<majority ratio>

the other hand, SSP appeared to strike a balance between BAP and BSP. This is expected since SSP is regarded as a compromise between strict synchronization (BSP) and full relaxation (BAP).

Looking at Fig. 7, fully relaxed BAP performed the least number of iterations (i.e., training rounds) by far due to its relaxed sync barriers. Naturally, partially relaxed BAP managed to perform significantly more rounds, eventually leading it to score the highest accuracy of all schemes. On the other hand, BSP had the furthest progression, managing to score very close to the top accuracy due to its faster pace. SSP performed fewer training iterations than BSP since it applies more relaxation on the sync barrier. However, SSP still outperforms BSP in terms of accuracy due to its better utilization of the training power. By and large, each one of these schemes suits certain system configurations (and purposes), depending on several factors, but the primary deciders are trainer pace and quality.

5.5 Experiment 4: Decentralized vs centralized performance for bayesian networks

In this experiment, we compare the performance of the centralized Bayesian networks model with that of the decentralized model, utilizing the best-performing model (as demonstrated later in experiment 5) comprising nine trainers and a single validator. The main goal of the experience is to test whether the advantages of the decentralized model can result in any degradation of the model quality.

5.5.1 Setup

In this experiment, we trained a centralized Bayesian network model using the whole training dataset. The decentralized model comprised nine trainers and a single validator, as it is the best-performing decentralized configuration. The training phase of the centralized and decentralized models ran for only one iteration, as calculating the CPT scores requires only one iteration.

5.5.2 Results and discussion

As suggested in table 2, the performance of the Decentralized model is slightly higher than the centralized model. Thus, the added benefits of decentralization did not result in performance degradation. This is not meant to suggest that the decentralized model performs better than the centralized model. It only means that decentralization does not result in any performance degradation.

Table 2 Scores of the centralized vs the decentralized model

Model	Accuracy
Centralized model	0.8383
Decentralized model	0.8386

5.6 Experiment 5: Trainers-to-validators ratio in bayesian networks

In this experiment, we want to see the best trainers-to-validators ratio for Bayesian networks. As stated in experiment 1, the trainers-to-validators ratio is a critical factor for the model performance and the usage of computing resources. We had to repeat this experiment since the nature of the Convolutional Neural Network (CNN) used in experiment 1 is greatly different from the nature of the Bayesian network model that we will test in this experiment. One key difference is that while the CNN trainers share their updated weights with the validators, the Bayesian Networks will send its calculated CPT scores. We will have a maximum of N participating nodes in the training and validation. We will try different combinations of the number of validators and the number of trainers under the condition that each trainer should send its CPT scores to exactly one trainer, which means that the number of trainers should be greater than or equal to the number of validators.

5.6.1 Setup

In this experiment, we set the N number of participating nodes to 10. In the initial run, we started with nine trainers and one validator. Then, we had eight trainers and two validators until we had five trainers and five validators. The experiment will last for only one iteration as calculating the CPT scores requires only one iteration. We will record the accuracy for each run to study what trainers-to-validators ratio yielded the best performance.

5.6.2 Results and discussion

From Fig. 8, we can see that there are no significant differences between the performances of the different splits. However, we can see that the best-performing configuration for this model (in terms of accuracy) is one with nine trainers and a single validator.

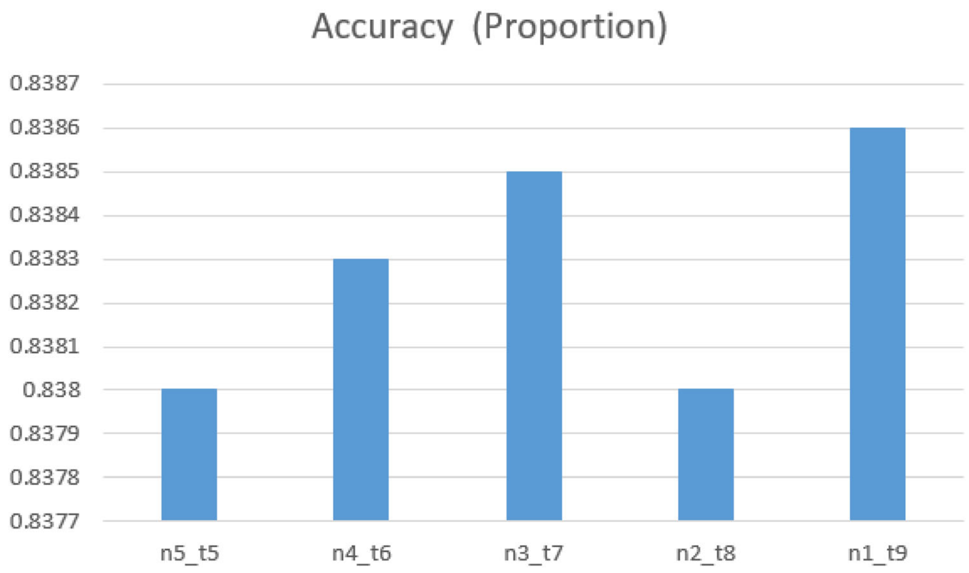


Fig. 8 Accuracy performance for different splits of trainers and validators

Table 3 Trust Scores of the different trainers

Trainer	Trainer 0	Trainer 1	Trainer 2	Trainer 3	Trainer 4	Trainer 5
Trust score	0.2608	0.2608	0.2534	0.1165	0.0697	0.0385

Table 4 Scores of the centralized vs the decentralized model having untrustworthy trainers

Model	Accuracy
Centralized model	0.8383
Decentralized model	0.838

Table 5 Comparative Results of FloBC against the different federated learning methods in [12]

Method	Accuracy
Local	86.48 ± 0.2
FedAvg	54.53 ± 0.6
FedAvg-FT	84.96 ± 0.2
Dis-PFL	91.05 ± 0.2
FLoBC	90.35 ± 0.2

5.7 Experiment 6: Reward-penalty policy in bayesian networks

The goal behind this experiment is to see to which extent calculating a trust score for each Bayesian network trainer helps optimize model performance if some trainers are not doing well in training.

5.7.1 Setup

In this experiment, we had 6 (indexed from 0 to 5) trainers and three validators. Trainers 0, 1, and 5 calculated their CPT scores normally from the dataset. The other three trainers (3,4,5) calculated their CPT scores randomly without using the training data. The purpose of having these trainers is to see whether the system can handle models whose scores are not accurate. Next, the trust scores of each of the trainers were recorded, and the accuracy of the aggregated was compared to the accuracy of the centralized model.

5.7.2 Results and discussion

From Table 3, we can see that the first three trainers, which got their scores from the data, had the highest trust scores and nearly dominated the aggregation of the CPT scores. Thus, our system is capable of detecting the trainers who are not trustworthy and limiting their contribution to the CPT scores aggregation.

From Table 4, it is clear that the trainers who generated random CPT scores did not affect the model performance. Thus, our system can tolerate trainers who generate CPT scores that do not resemble the actual probabilities in the dataset. From a cyber security angle, if a cyber attack happened on one of the trainers, causing them not to be accurate, the system can still tolerate them and maintain a good performance.

5.8 Experiment 7: Comparing FLoBC to Dis-PFL

In this experiment, we compare our method with Dis-PFL(Decentralized sparse training based Personalized

Table 6 Results for the 3 x (128 FC) model architecture

Method	1 epoch	5 epochs	10 epochs
PVD-FL	91.49	96.84	97.37
FLoBC	94.48	96.43	97.13

Table 7 Results for the 3 x (512 FC) model architecture

Method	1 epoch	5 epochs	10 epochs
PVD-FL	94.63	97.54	97.94
FLoBC	95.30	97.60	97.96

Federated Learning), which is a decentralized federated learning framework that employs personalized sparse masks and a decentralized sparse training technique to customize local models on the edge [12]. We chose to utilize CFAIR 10 dataset on ResNet18 model that was used in [12] to reproduce the original results across all baselines and compare them with our proposed framework.

5.8.1 Setup

In this experiment, the dataset is pathologically partitioned across trainers which means that each trainer is given only a limited set of classes to work on. We set the number of classes per trainer to be two classes to resemble the binary classification in the original experiment, with a total of 100 trainer nodes. Since each trainer works only on two classes, the updates of this trainer has to be sent to a validator that had a validation set consisting only of those two classes. As a result, for every possible combination of two classes, we created a corresponding validator, resulting in 45 validator nodes.

5.8.2 Results and discussion

As shown in table 5, we can see that our proposed framework (FLoBC) surpassed all federated learning baselines, and it provided very close performance to Dis-FPL method. This result shows that FLoBC can actually handle data with completely different distributions from multiple trainers, with pluggable machine learning models for training, and providing comparable performance to state of the art specialized models.

5.9 Experiment 8: Comparing FLoBC to PVD-FL

In this experiment, we compare our method with another state of the art federated learning method which is PVD-FL(privacy-preserving and verifiable decentralized federated learning framework). PVD-FL is a decentralized federated learning framework, aiming to overcome security challenges in federated learning by introducing an efficient and verifiable cipher-based matrix multiplication algorithm (EVCM) and decentralized algorithms [14]. We choose the experiment the authors of this papers carried out using the MNIST dataset and different machine learning models architectures.

Table 8 Results for the CNN model architecture

Method	1 epoch	5 epochs	10 epochs
PVD-FL	96.54	97.54	98.47
FLoBC	96.56	98.06	98.58

5.9.1 Setup

We followed the setup of the experiment in [14], with three different model architectures which are a CNN (32 Cov 5 $\rightarrow$ MP 2 $\rightarrow$ 64 Cov 5 $\rightarrow$ MP 2 $\rightarrow$ 512 FC $\rightarrow$ 10 CEE), two FCNs (Fully connected networks) with architectures: 3 $\times$ (128 FC) $\rightarrow$ 10 CEE, and 3 $\times$ (512 FC) $\rightarrow$ 10 CEE. The learning rate is set to 1, and the batch size is set to 32, and four trainer nodes. We used two validators for FLoBC setup.

5.9.2 Results and discussion

As shown in Tables 6, 7, and 8, we can see that FLoBC provides very close accuracy to PVD-FL for one model architecture, and provides better accuracy on the other two model architectures. We can also note that the convergence rate for FLoBC is higher relative to the compared baselines.

6 Conclusion

In this paper, we presented a novel, decentralized, privacy-preserving federated learning framework to tackle the problem of centralized training of machine learning models, which requires high computational power and, more importantly, huge storage, sometimes practically infeasible. We show that our proposed framework achieves the desired levels of generality, decentralization, efficiency, privacy, in addition to Byzantine fault-tolerance. Through our benchmark experiments, our decentralized learning framework exhibits itself to be a viable match to classic centralized training models, marginally outperforming the centralized benchmark by a factor of 0.5% in case of SGD-based models. Moreover, it matches the performance of a centralized model using bayesian optimization. In addition, we show that there is a quasi-optimal balance to strike on the trainer-to-validator ratio, empirically determining that a 7:3 trainer-to-validator ratio works best, for the data sets used and the experiments conducted. Furthermore, we demonstrate that even a simple reward-penalty policy can have a notable positive effect on the quality of the produced federated learning models. Finally, we compare three major synchronization schemes (BSP, SSP, & BAP), highlighting and contrasting their performance gains and trade-offs.

7 Future work

For future work, we plan on investigating further improvements that can be applied to the synchronization schemes for our framework by allowing more adaptability in the sync periods based on the expected time that most trainers take to submit their updates in one round. Furthermore, to ensure a higher level of data privacy, we propose to test the introduction of a layer of differential privacy into transaction communications to limit the extent to which the gradients can be exploited to extract the trainers' data.

Author contributions Y.A. was responsible for the framework extension to include bayesian optimization with the corresponding experiments and its writeup, M.G., H.G, E.S. and F.D. worked on the original gradient optimization and block-chain infrastructure and the corresponding experiments and their writeup, H.S. and T.E. wrote parts of the manuscript text, and all authors reviewed the manuscript

Funding Open access funding provided by The Science, Technology & Innovation Funding Authority (STDF) in cooperation with The Egyptian Knowledge Bank (EKB).

Declarations

Competing interests The authors declare no competing interests.

References

1. McMahan, B., Moore, E., Ramage, D., Hampson, S., Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: Artificial Intelligence and Statistics, pp. 1273–1282 (2017). PMLR
2. Mao, Q., Wang, L., Long, Y., Han, L., Wang, Z., Chen, K.: A blockchain-based framework for federated learning with privacy preservation in power load forecasting. Knowl.-Based Syst. **284**, 111338 (2024). https://doi.org/10.1016/j.knosys.2023.111338
3. Ghanem, M., Dawoud, F., Gamal, H., Soliman, E., ElBatt, T., Sharara, H.: Flobc: a decentralized blockchain-based federated learning framework. In: 2022 Fourth International Conference on Blockchain Computing and Applications (BCCA), pp. 85–92 (2022). IEEE
4. Zheng, Z., Xie, S., Dai, H.-N., Chen, X., Wang, H.: Blockchain challenges and opportunities: a survey. Int. J. Web Grid Serv. **14**(4), 352–375 (2018)
5. Lu, Y., Huang, X., Dai, Y., Maharjan, S., Zhang, Y.: Blockchain and federated learning for privacy-preserved data sharing in industrial IoT. IEEE Trans. Ind. Inform. **16**(6), 4177–4186 (2019)
6. Qu, Y., Uddin, M.P., Gan, C., Xiang, Y., Gao, L., Yearwood, J.: Blockchain-enabled federated learning: a survey. ACM Comput. Surv. **55**(4), 1–35 (2022)
7. Nguyen, D.C., Ding, M., Pham, Q.-V., Pathirana, P.N., Le, L.B., Seneviratne, A., Li, J., Niyato, D., Poor, H.V.: Federated learning meets blockchain in edge computing: opportunities and challenges. IEEE Internet Things J. **8**(16), 12806–12825 (2021)
8. Gibiansky, A.: Bringing HPC techniques to deep learning. Baidu Research, Tech. Rep. (2017)
9. Li, H., Kadav, A., Kruus, E., Ungureanu, C.: Malt: distributed data-parallelism for existing ML applications. In: Proceedings of the Tenth European Conference on Computer Systems, pp. 1–16 (2015)
10. Agarwal, A., Chapelle, O., Dudík, M., Langford, J.: A reliable effective terascale linear learning system. J. Mach. Learn. Res. **15**(1), 1111–1133 (2014)
11. Lalitha, A., Kilinc, O.C., Javidi, T., Koushanfar, F.: Peer-to-peer federated learning on graphs (2019). arXiv:1901.11173
12. Dai, R., Shen, L., He, F., Tian, X., Tao, D.: Dispfl: towards communication-efficient personalized federated learning via decentralized sparse training. In: International Conference on Machine Learning, pp. 4587–4604 (2022). PMLR
13. Hegedűs, I., Danner, G., Jelasity, M.: Gossip learning as a decentralized alternative to federated learning. In: IFIP International Conference on Distributed Applications and Interoperable Systems, pp. 74–90 (2019). Springer, Berlin
14. Kumar, S., Joshith, T., Lokesh, D., Dasari, J., Mahato, G., Chakraborty, S.: Privacy-preserving and verifiable decentralized federated learning. In: 2023 5th International Conference on Energy, Power and Environment: Towards Flexible Green Energy Technologies (ICEPE), pp. 1–6 (2023). https://doi.org/10.1109/ICEPE57949.2023.10201599
15. Li, Y., Chen, C., Liu, N., Huang, H., Zheng, Z., Yan, Q.: A blockchain-based decentralized federated learning framework with committee consensus. IEEE Netw. **35**(1), 234–241 (2020)
16. Kim, H., Park, J., Bennis, M., Kim, S.-L.: Blockchained on-device federated learning. IEEE Commun. Lett. **24**(6), 1279–1283 (2019)

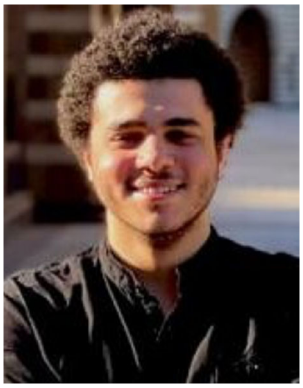
17. Pokhrel, S.R., Choi, J.: Federated learning with blockchain for autonomous vehicles: analysis and design challenges. IEEE Trans. Commun. **68**(8), 4734–4746 (2020)
18. Martinez, I., Francis, S., Hafid, A.S.: Record and reward federated learning contributions with blockchain. In: 2019 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), pp. 50–57 (2019). IEEE
19. Fadaeddini, A., Majidi, B., Eshghi, M.: Secure decentralized peer-to-peer training of deep neural networks based on distributed ledger technology. J. Supercomput. **76**, 1–15 (2020)
20. Pokhrel, S.R., Choi, J.: A decentralized federated learning approach for connected autonomous vehicles. In: 2020 IEEE Wireless Communications and Networking Conference Workshops (WCNCW), pp. 1–6 (2020). IEEE
21. King, S., Nadal, S.: Ppcoin: peer-to-peer crypto-currency with proof-of-stake. self-published paper, August **19**(1) (2012)
22. Verbraeken, J., Wolting, M., Katzy, J., Kloppenburg, J., Verbelen, T., Rellermeyer, J.S.: A survey on distributed machine learning (2019). arxiv:1912.09789
23. Kairouz, P., McMahan, H.B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A.N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., et al.: Advances and open problems in federated learning (2019). arXiv:1912.04977
24. Thapa, C., Chamikara, M.A.P., Camtepe, S.: Splitfed: when federated learning meets split learning (2020). arXiv:2004.12088
25. Bottou, L.: Large-scale machine learning with stochastic gradient descent. In: Proceedings of COMPSTAT'2010: 19th International Conference on Computational StatisticsParis France, August 22–27, 2010 Keynote, Invited and Contributed Papers, pp. 177–186 (2010). Springer, Berlin
26. Yanovich, Y., Ivashchenko, I., Ostrovsky, A., Shevchenko, A., Sidorov, A.: Exonum: Byzantine fault tolerant protocol for blockchains. bitfury. com, 1–36 (2018)
27. Bottou, L., Cortes, C., Denker, J.S., Drucker, H., Guyon, I., Jackel, L.D., LeCun, Y., Muller, U.A., Sackinger, E., Simard, P., Vapnik, V.: Comparison of classifier methods: a case study in handwritten digit recognition. In: Proceedings of the 12th IAPR International Conference on Pattern Recognition, Vol. 3—Conference C: Signal Processing (Cat. No.94CH3440-5), vol. 2, pp. 77–822 (1994). https://doi.org/10.1109/ICPR.1994.576879
28. Beinlich, I.A., Suermondt, H.J., Chavez, R.M., Cooper, G.F.: The alarm monitoring system: a case study with two probabilistic inference techniques for belief networks. In: AIME 89: Second European Conference on Artificial Intelligence in Medicine, London, August 29th–31st 1989. Proceedings, pp. 247–256 (1989). Springer, Berlin

Youssif Abuzied senior computer engineering student in the American University in Cairo. He is interested in research, especially machine learning. He has several work experiences as an undergraduate teaching assistant and as a research assistant.

Mohamed Ghanem Doctoral Researcher at the CISPA Helmholtz Center for Information Security (Reactive Systems Group), pursuing my PhD at Saarland University. He primarily interested in Neuro-Symbolic Artificial Intelligence, Deep Learning, and Computer Vision. He obtained his Master's degree from Saarland University (grade: excellent) in Saarbrücken. Earlier, He graduated from The American University in Cairo with a Bachelor's degree in Computer Engineering (Highest Honors) and a minor in Mathematics.

Fadi Dawoud An alumnus of the American University in Cairo, graduated as valedictorian in 2021 with a B.Sc. in Computer Engineering and a minor in Mathematics. Presently, he contributes to high-visibility projects at Microsoft, with a keen interest in software engineering, data analytics, and quantum computing.

Habiba Gamal is a backend and infrastructure software development engineer at Microsoft. She is interested in exploring & implementing best practices to develop scalable, maintainable and highly available web applications. She currently works on the notifications platform that serves millions of notifications per day on different Microsoft canvases. Habiba holds a Bachelor's of Science in Computer Engineering, graduating Summa Cum Laude, with a minor in Business Administration from the American University in Cairo. During her undergraduate studies, she was focusing her research on federated learning and electronic design automation.



Eslam Soliman currently serving as a Software Development Engineer 2 at Microsoft. He graduated from the American University in Cairo with a bachelor's degree in 2021. His focus primarily lies in web development, where he specialize in crafting seamless user experiences and scalable solutions. Throughout his career, he had the privilege of contributing to the delivery of new user-facing features that have positively impacted millions of users worldwide. He expertise spans various aspects of software engineering best practices, with a particular interest in DevOps. While his deeply passionate about leveraging automation to enhance development workflows and product quality, his core strength lies in web development. He take pride in ability to tackle challenges in this domain and deliver high-quality solutions.

Hossam Sharara As an Assistant Professor, Hossam Sharara joined the Department of Computer Science and Engineering at The American University in Cairo (AUC) in July 2018. He got his BSc and MSc degrees in computer science and engineering from Alexandria University, Egypt, in 2004 and 2007, respectively. He then joined the Department of Computer Science at the University of Maryland, College Park, USA as a research assistant in the year 2007, from where he obtained another MSc degree in 2010 and received his PhD degree in relational machine learning and data ,ining in 2012 under the supervision of Prof. Lise Getoor. His PhD research was partially supported by the Dean's fellowship award, which he was granted from the University of Maryland, College Park, in 2010. From 2012 to 2018, Sharara worked as a senior engineer/research scientist at Google (2012 - 2016), and Facebook (2016 - 2018). He returned to Egypt in early 2018, joining Uber Egypt as a Senior Manager of Business and Data Analytics. Sharara has authored and co-authored numerous publications that appeared in top-tier, peer-reviewed conferences and journal proceedings, including ACM SIGKDD Conference on Knowledge Discovery and Data Mining, AAAI Conference on Weblogs and Social Media (ICWSM), International Joint Conference on Artificial Intelligence (IJCAI), Journal of Advances in Social Networks Analysis and Mining, and others. In addition, he co-authored a book chapter in "Link Mining: Models, Algorithms and Applications" by Philip S. Yu, Christos Faloutsos, and Jiawei Han and an article in the Encyclopedia of Machine Learning. He also served on the program committee at several data mining conferences, as a member of the NSF review panel on Graph Mining in 2013 and on the academic review panel at Google.

Tamer ElBatt has been a Professor at the CSE Dept., The American University in Cairo (AUC) since 2017. He received the B.S. and M.S. degrees in EECE from Cairo University, Egypt in 1993 and 1996, respectively, and the Ph.D. degree in ECE from the University of Maryland, College Park, USA in 2000. From 2000 to 2009 he was with HRL Labs, USA and Lockheed Martin ATC, USA, at various positions. From 2009 to 2017, he served at the EECE Dept., Cairo University. He also held a joint appointment with Nile University, Egypt from 2009 to 2017 and served as the Director of the wireless research center (WINC) from 2012 to 2017. At the AUC, he served as the CSE Graduate Director and then Associate Chair, from 2019 to 2022. He has published more than 145 papers in major journals and international conferences. Dr. ElBatt served on the TPC of numerous IEEE and ACM conferences. He currently serves on the Editorial Board of Frontiers in Communications and Networks – Data Science for Communications and has served on the Editorial Board of IEEE TCCN, TMC and Wiley IJSCN. Dr. ElBatt is an IEEE ComSoc Distinguished Lecturer, a recipient of the Google Faculty Research Award in 2011, among other national awards. His research interests lie in the broad areas of performance analysis, protocol design and optimization of wireless networks, mobile computing and IoT. He served as the IEEE Egypt Section Vice Chairman (2020-2023). Dr. ElBatt is a Senior Member of IEEE.