Federated Learning using Smart Contracts on Blockchain, based on Reward Driven Approach

Monik Raj Behera, Sudhir Upadhyay, Suresh Shetty
{monik.r.behera,sudhir.x.upadhyay,suresh.shetty}@jpmorgan.com

Abstract—**Over the recent years, Federated machine learning continues to gain interest and momentum where there is a need to draw insights from data while preserving the data provider's privacy. However, one among other existing challenges in the adoption of federated learning has been the lack of fair, transparent and universally agreed incentivization schemes for rewarding the federated learning contributors. Smart Contracts on a Blockchain network provide transparent, immutable and independently verifiable proofs by all participants of the network. We leverage this open and transparent nature of smart contracts on a blockchain to define incentivization rules for the contributors, which is based on a novel scalar quantity - federated contribution. Such a smart contract based reward-driven model has the potential to revolutionize the federated learning adoption in enterprises. Our contribution is two-fold: first is to show how smart contract based blockchain can be a very natural communication channel for federated learning. Second, leveraging this infrastructure, we can show how an intuitive measure of each agents' contribution can be built and integrated with the life cycle of the training and reward process.**

Index Terms—**Machine Learning, Blockchain, Federated Learning, Fairness in Federated Learning, Systems infrastructure for Federated Learning**

I. Introduction

THE concept of **federated machine learning** was introduced around 2016[1]. It relies on the principle of remote and distributed execution of machine learning algorithm, and the ability to share and aggregate individual models in a secure and anonymous manner. Therefore, it is implicit that federated machine learning would depend on availability of secure communication channels between remote participants to allow distribution of locally trained individual models.

Blockchain became popular with launch of bitcoin around 2009[2]. Blockchain is a form of distributed ledger technology that relies on honest majority members in a network to validate the accuracy of the executed transactions on the network. It accomplishes this by allowing each of its members to execute a piece of turing complete software code (a.k.a smart contract), in an independent fashion without any external influence or interventions. Although the proposed solution could be extended to other Blockchains, this paper focuses primarily on Ethereum's implementation. Therefore, Blockchains can help the deployment of federated learning by both bringing dataset on a unique, structured, ledger (with potential privacy layers on it), and by guaranteeing the security, accuracy and correctness of the distribution of the model's parameters. This could be of particular value in compliance and anti-money laundering cases requiring the reconciliation of multiple sensible dataset, and in which the use of fraud and anomaly detection models could improve manual audits and investigations[3].

However, because of the distributed nature of validation, in current blockchains, unless it implements a privacy layer, all communications between any two nodes is visible to rest of the nodes of the network. Preserving privacy of transaction in a blockchain, while still allowing all nodes to participate in consensus process is a difficult problem to solve. This is an active area of research, and includes technologies such as Zero Knowledge Proof (*zKP*). The foundations of *zKP* is based on interactive proofs, as described in previous works[4] [5] [6]. However, setup of those continues to be an onerous process, in real world implementation.

This poses a challenge for federated learning, that requires maintaining the privacy of the individual participant's machine learning models (and model gradients too) and anonymity of the model contributor. The proposed implementation solves this challenge via dynamically generating asymmetric and symmetric keys for each federated learning round (details to follow in subsequent sections), with a caveat that the aggregation server node is conceptually akin to a Trusted Execution Environment (TEEs), but at a consortium level[7].

However, even with consortium trusted aggregation server implementation, the risk of lack of contribution in the overall federated learning rounds from individual nodes continues to exist. In addition, the malicious nodes could potentially send a misleading model that could skew the efficacy of the aggregated models. One of the potential ways to address the above challenges could be to have the aggregation server detect somehow, such behavior and drop those contributors from the collaboration process. Unfortunately, this solution tends to centralizes the solution and lack of transparency in the overall process.

The current paper proposes a solution to avoid the above potential pitfalls by leveraging unique and transparent smart contracts design on blockchain to reward honest/active (and penalize malicious/under performing) participants in the learning process, based on computing a novel, scalar quantity - **federated contribution**. In our proposed solution, smart contract is responsible for reward (or penalty) specification and distribution (or fees) as well, through immutable federated contribution records on blockchain.

Monik Raj Behera, Onyx Engineering, J.P.Morgan Chase & Co, India (e-mail: monik.r.behera@jpmorgan.com).

Sudhir Upadhyay, Onyx Engineering, J.P.Morgan Chase & Co, India (e-mail: sudhir.x.upadhyay@jpmorgan.com).

Suresh Shetty, Onyx Engineering, J.P.Morgan Chase & Co, India (e-mail: suresh.shetty@jpmorgan.com).

II. Related work

A. Federated Learning

Federated learning is a distributed machine learning setting where the goal is to train a high quality global model, while training is done locally and privately on individual participant (federated learning client). The training is done across large number of clients. After the local models are trained, the improved local model gradients are sent securely over the network to the federated server for federated aggregation[8]. This paper is focused primarily on horizontal federated learning.

B. Blockchain

There are several implementations of blockchain - Bitcoin, Ethereum, Hyperledger Fabric etc. Ethereum[9] is a decentralized, open-source blockchain with smart contract functionality. Ether is the native cryptocurrency of the platform. After Bitcoin, it is the second-largest cryptocurrency by market capitalization. In addition to existing public Ethereum mainnet, there exists Ethereum for Enterprises[10] that offer certain enterprise desired features. Some of these features include more control on nodes in the network, higher performance, difference in cost, node permissioning and different privacy implementation.

In general, there are three main types of Blockchains[11]

- **Public** Blockchain: a blockchain that anyone in the world can read, anyone in the world can send transactions to and expect to see them included if they are valid, and anyone in the world can participate in the consensus process.
- **Consortium** Blockchain: a consortium blockchain is a blockchain where the consensus process is controlled by a pre-selected set of nodes.
- **Fully private** Blockchains: a fully private blockchain is a blockchain where write permissions are kept centralized to one organization. Read permissions may be public or restricted to an arbitrary extent.

Some implementations of Blockchain, such as Ethereum-support both public and Consortium implementations. One-such enterprise/consoritum version of Ethereum is Consensys Quorum . The proposed solution in this paper applies to the **Consortium** (Enterprise) blockchain where the identity of individual nodes are known to the Consortium Operator (a.k.a. Network Operator).

There are few key components of a Blockchain eco-system that are relevant to this paper. Those include namely Smart Contract, Events and Oracles.

C. Smart Contract on Blockchain

A **smart contract**[12] is simply a program that runs on the Ethereum blockchain. It's a collection of code (its functions) and data (its state) that resides at a specific address on the blockchain. Smart contract are permission-less, i.e. anyone can write a smart contract and deploy it to the network. The smart contract code is visible to all participants on the network, and any participant can independently execute that code to validate the outcome[13]. Smart Contracts on Ethereum are written in its own programming language called **Solidity**. **Events** on Ethereum[14] are well defined ways of asynchronously exchanging data among the participants of blockchain network. In Solidity, events[15] are dispatched as signals, that the smart contracts can fire. *DApps*, which are essentially decentralized applications, or anything connected to Ethereum JSON-RPC API (an interface exposed by blockchain network for connectivity and programmatic interactions with blockchain), can listen to these events and act accordingly. Event can also be indexed, so that the event history can be searched later.

D. Events on Blockchain

In Blockchain, when a transaction is mined, smart contracts can emit events and write logs to the blockchain that the frontend can then process. These events can then be used to communicate with a smart contract from application frontend or other subscribing applications. Events are not considered as a state change on Blockchain, hence they consume very less gas price[16], in comparison to state change transactions on Blockchain.

E. Federated learning and Blockchain

There is a growing literature on federated learning implementations through blockchain, indicating a sign of the natural complementarity between these two technologies. Since [17] proposed using blockchain to maintain the global model with community and reach a consensus, a number of papers [18], [19], [20], [21] explored this avenue, but mainly using the blockchain as a safe and coherent storage for the global model, and fail to make full use of the potential of smart contracts to both coordinate the learning, and through that compute measurement functions of how each agent is contributing to the global model.

For that measurement of contribution framework we build on [22] proposal to leverage the blockchain to evaluate updates from nodes, and potentially penalize malicious nodes. Shapley values have shown great results in explaining the contributions of individual features in theoritically any machine learning model. It is our hope to further bridge these two literatures, to be able to automatically compute different variations of federated learning contributions through blockchain-based smart contract as communication medium in federated learning settings. Our main contribution is to showcase how a natural infrastructure and life cycle could support these, leveraging the cryptographic, distributed computing, and consensus mechanisms within blockchain.

F. RSA and AES algorithm

Asymmetric key cryptographic algorithm are popular cryptography techniques, which focus on using public-private key pair for encryption. In RSA algorithm[23], the fundamental idea is to use a computationally impossible, long prime numbers based public key and private key. Public key can be used to encrypt data, where as it can only be decrypted using private key, which is kept privately and securely with the owner. In case of symmetric cryptography, like of AES algorithm[24], the same key would be used for encryption and decryption. Thus special care needs to be given to safeguard the symmetric key itself.

G. Measuring contributions in federated learning

In recent work[25], measurement of contributions towards improvising the global model in federated learning has been described, for both horizontal and vertical class of federated learning. Authors have discussed the approach of using 'deletion method' for horizontal federated learning approach, where the change in testing accuracy is considered by various iterations of training, with each iteration, we remove the data points from a single client. This way, we measure the degradation in model performance, and then infer the contributions accordingly by the federated clients. In case of vertical federated learning, Shapley values are computed for each feature. Shapley values gives a strong quantification of each feature's importance, followed by a mathematical approach of inferring contributions of parties in vertical federated learning. There are, however, a multiplicity of ways in which the Shapley value can be implemented, with very disparate results, as shown in [26]. Our contribution aims to build a mathematical foundation to compute contribution of participants in federated network by focusing on weights differences using Frobenius Norms, instead of training data differences like in Shapley values.

III. BLOCKCHAIN FOR FEDERATED COMMUNICATION

Earlier works have proved *REST, RPC, gRPC*[27] as popular choices for communication between federated server and clients. For more secure communication, various battle-tested approaches include network firewalls, SSL and token based authentication. Though these methods provide a secure medium, they lack the transparency in the communication channel itself, which can be well established by using blockchain. This paper proposes use of smart contracts on (*ethereum based*) Blockchain, which provides a decentralized mechanism of communication between peers and the federated server. While smart contracts provide transparency in the implementation, the communications at blockchain level are secured via asymmetric cryptography techniques for encryption and modification of state on the blockchain network. This also entrusts the emission of events for communication between federated server and contributor nodes with extended level of security and transparency.

Owing to the "consortium blockchain network", where individual participants are trusted enterprises and organizations, there is a likelihood of participants acting as a *bad* actors is low. Further, in this network of contributors, it is quite possible that there could be wide range of variation in contributions from different participants or even extreme scenario of some participants being only a benefitor, and not contributor (and vice versa). These are few of the possible scenarios in consortium governed federated learning network. To manage and counter these measures through incentives *(both reward and penalty)*, smart contracts provide a transparent and immutable way of maintaining contributions record on the blockchain. Based on the contribution record from blockchain, a given participant would be either rewarded or penalized through on-chain blockchain tokens, or through off-chain mechanism established by consortium.

A. Performance Characteristics of Blockchain

A well-known constraint of existing blockchain implementations are around performance, usually measured in throughput and latency. Further, current Blockchain implementations also suffer from Blockchian Trilema (a coin termed by Vitalik Buterin) - speed, security and decentralization. These constraints are quite true and applicable for high volume, low latency networks where speed of transaction is quite critical. However, sharing models over blockchain in an enterprise network does not require significant high throughput sine the frequency of model updates is expected to be relatively low. In addition, latency is also not a critical factor since aggregation of models from each participant are not necessarily time sensitive. For example, if a model update was missed by the aggregation server in one aggregation cycle, it will be picked up in the subsequent one without losing its impact.

B. Life-cycle of a federated aggregation event

Federated learning in a consortium network comprises of a consortium trusted and security hardened aggregation server. All the participants, who are clients in the federated learning ecosystem listen to the blockchain events from a smart contract (*ethereum smart contracts, as ethereum is the blockchain network being used*). Table I gives an overview of a typical federated learning round. In the proposed implementation, the Aggregator Server **manages** the events that orchestrate the Federated Learning cycle between clients and aggregator.

Since the orchestration of events is delegated to the aggregator, this design allows individual participants remain lightweight and only focus on local model improvisation. The set of possible events from Aggregator server includes initial distribution of base model, subsequent federated learning cycle events, contribution announcement on the chain and contribution fees notification.

C. Encryption of event data

In the previous sections, we discussed about how blockchain ensures required security and privacy at its core. Though data on blockchain is immutable, the privacy is not guaranteed, because of the very nature of how blockchain works. A consortium network with enterprise participants, there is a high possibility of participants not comfortable with revealing their model weight gradients to peer participants. If local model weights (*and gradients*) are revealed, this posses risk of revealing statistical properties of the data from individual participant, if not the actual data.

In order to privately send model weights from participant to federated server on blockchain, in a consortium network, asymmetric encryption using **RSA** cryptography is proposed in the paper. Each new federated learning round is published as blockchain event, federated aggregation server generates a new set of RSA key pair. The private key of the pair stays with

TABLE I
LIFE-CYCLE OF A FEDERATED LEARNING COMMUNICATION EVENT. TIME COLUMN REPRESENTS RELATIVE TIMESTAMP, WHERE INCREMENT IN TIMESTAMP SUGGESTS RELATIVE INCREMENT, NOT THE ABSOLUTE INCREMENT. EVENT IS THE RESPECTIVE EVENT TYPE ON COMMUNICATION ROUND. 'ACTION ON' COLUMN SHOWS THE EXPECTED ACTION FOR WHICH ROLE, WHETHER CLIENT OR SERVER. 'PUBLISH' COLUMN DEPICTS WHETHER THE EVENT IS PUBLISHED AS BROADCAST ON BLOCKCHAIN OR NOT.

TIME	EVENT	ACTION ON-	PUBLISH?
t_k	INITIATE FL ROUND	SERVER	$\surd$
t_{k+1}	RECEIVE INITIATE EVENT	CLIENT(S)	$\times$
t_{k+2}	ENCRYPT LOCAL MODEL; WITH KEY RECEIVED IN PREVIOUS STEP	CLIENT(S)	$\times$
t_{k+3}	PUBLISH ENCRYPTED LOCAL MODEL	CLIENT(S)	$\surd$
t_{k+4}	AGGREGATE ENCRYPTED LOCAL MODELS	SERVER	$\times$
t_{k+5}	BROADCAST NEW GLOBAL MODEL	SERVER	$\surd$

the server, as this will be used by the server in later stages to decrypt the encrypted event message cipher. The public key is sent across the network to all the participants. Participant will generate an AES key for encrypting local train model, and then use the public key received from federated server to encrypt the AES key[28]. With RSA keys revolving for every new federated learning round, this decreases the chances of compromising model information over the blockchain, for any given participant on the network. One thing to note here is - only the the event data containing participant's local model weights needs to be encrypted. Other event data like global model, which is sent from server to clients, initiation of federated round broadcast does not require encryption, as it does not contain sensitive data among the peers in blockchain network.

D. Smart Contract for communication and contribution towards Federated Learning

In the previous sections, smart contracts have been described as a way of executing Turing complete programs on blockchains. Further, as referred in related work, Ethereum smart contracts also have a concept of events, which are generated after a transaction is mined on the Blockchain network. Events can carry additional information and parameters that can be used by subscribing applications. It is important to understand that events generated, when published by the participants or the federated server, would be a broadcast. Dynamically generated encryption keys, explained in earlier section would safeguard the privacy interest of the federated clients. Federated clients would have to listen to the agreed upon Ethereum *event* from a smart contract, in order to receive the event and take necessary actions, essentially making smart contracts a primary *pub-sub*[29] channel of communication.

In the current paper, smart contracts are proposed as a mechanism to maintain transparent, immutable records of **contributions**, which improvises (*or degrades*) the global federated learning model. The computation to determine the contribution of individual in the federated learning round (*described in subsequent section*) is performed off the blockchain, considering the resource and computation constraints. After the computation is completed, the records are published on blockchain, **anonymously**, for each participant to consume. This makes the contributions (and indirectly participants expense fees) transparent. The transparency of contributions from each participant promotes an honest behavior on the network.

IV. FEDERATED AGGREGATION AND CONTRIBUTION

Federated learning with centralized aggregation server implements various ways to implement federated aggregation[30], like *FedAvg*, *FedSGD*, etc. Assuming the number of clients is quite large (*in order of* 10^5), if few of the clients are acting as *bad* actors, or clients with noise in training data, their impact towards the global model is smoothed by averaging algorithm. But in case of consortium setting, where number of clients is not that huge (*in order of* 10^2), the local weights of federated client with noisy data or malicious intention would impact the overall weight(s) of the federated learning model. In order to tackle the challenge, we are proposing a novel way of computing a novel scalar quantity, **federated contribution** across the network, and using smart contract to publish and store on the blockchain (as discussed in earlier section). As compared to earlier work[25], federated contribution establishes a way to define contribution of each participant in federated learning setting, whether they might have train data of similar statistical properties, or non overlapping (*orthogonal in feature space*) statistical properties.

A. Federated aggregation

FedAvg[31] is one of the popular algorithms in federated aggregation domain, which ensures complete and optimal solution, provided the learning rate and local learning contribution is accurately considered. Inspired from the previous works, our problem formulation for non-IID (*or IID*) data, which assumes more real world problem setup, in a consortium blockchain network, we have formulated our problem as described below -

$$\min_{\theta \in \mathbf{R}^{\mathbf{d}}} f(\theta) := \sum_{k=1}^{K} p_k F_k(\theta)$$

where k as index for client, θ as the set of model weight parameters for any generic machine learning algorithm, which can be modified for federated learning setting. $f(\theta)$ is the global objective function, $F_k(\theta)$ is the local objective function for client k, p_k is the learning importance factor - a scalar value to determine individual client's, local model's relative importance, and K is the total number of clients. Considering p_k as learning importance factor in federated learning, it is assumed that -

$$\sum_{k=1}^{K} p_k = 1$$

With *fedAvg* as algorithm for aggregation, the individual weight updates for $t+1$ iteration, l^{th} layer can be defined as -

$$w_{t+1,l} \leftarrow w_{t,l} + \alpha \sum_{k=1}^{K} p_k \nabla F_k(\theta)$$

where $w_{t,l}$ is weight for t^{th} iteration at l^{th} layer and α is the learning rate parameter.

B. Federated learning contribution

In previous sections, motivation for **federated contribution** is discussed. Intuitively, federated computation is a scalar quantity, which depicts the deviation, or divergence of two machine learning models. We will now define the federated contribution mathematically.

$$\gamma^k := \|\beta^k\|_2$$

$$\beta^k := \langle \|\delta_1^k\|_{\mathbf{F}}, \quad \|\delta_2^k\|_{\mathbf{F}}, \quad \dots \quad , \|\delta_L^k\|_{\mathbf{F}} \rangle$$

$$\delta_l^k := \mathbf{w_{l,t}^{global}} - \mathbf{w_{l,t+1}^{k}}$$

$$\gamma_{rel}^k := \frac{\gamma^k}{\sum_{k=1}^{K} \gamma^k}$$

where γ^k is the absolute federated contribution of client k, $\|.\|_p$ represents p^{th} norm, $\|.\|_F$ represents Frobenius norm[32], L represents the final layer's weight matrix of a generic machine learning model. δ_l^k represents difference of model weight parameter matrix for l^{th} layer of k^{th} client. $\mathbf{w_{l,t+1}^{k}}$ represents model weight for l^{th} layer of k^{th} client at $t+1^{th}$ iteration. $\mathbf{w_{l,t}^{global}}$ is the model weight for l^{th} layer of global model at t^{th} iteration. γ_{rel}^k is relative federated contribution of client k.

For any generic machine learning algorithm, like linear regression, logistic regression, neural networks, etc, the primary intent is to find a weight matrix, or a set of weight matrices. These weight matrices are computed using loss functions and gradient descent approaches, in various formats depending upon the algorithm. Assuming the representational vector, defined β^k, with minimum size of the set as 1, we can define an each element of β^k as Frobenius norm of δ_l^k. Here, δ_l^k, as defined previously, represents the element wise difference of weight matrices (order of subtraction doesn't matter, as Frobenius norm computes square of the element). We have considered the element wise difference of both the weight matrices for each layer, as it is inverse operation to the gradient descent and weight updates, where we perform addition of improvements, as described in problem formulation earlier. Frobenius norm is well established method to find the magnitude of a matrix from origin of a hyperspace $\mathrm{R^d}$. When we compute Frobenius norm for δ_l^k, it essentially represents the magnitude of deviation between two weight matrices(global weight matrix of previous iteration and local weight matrix of current iteration, for layer *l*). The vector β^k represents a set of magnitudes of deviation of local model's weight matrices from global model's weight matrices respectively, as an independent and individual axis. Finally, in order to calculate value γ^k for client *k*, we calculate *2-Norm*, which is the *euclidean norm* of β^k. This represents the magnitude of distance from origin, which quantifies the combined deviation of local model (*set of weights*) from global model (*again, set of weights*) as a scalar quantity.

Inferring γ^k, in case of federated contribution, federated aggregation server calculates γ^k for each k^{th} client. If the federated contribution value is relatively high, that means the given client has contributed to a *higher degree*. While describing contribution to a higher (*or lower*) degree in federated contribution, it practically quantifies the contribution in modifying the global model, by training on larger data points, or by training over data points which are having distant statistical properties from earlier training data, or may be higher noise in data. This intuitively can be thought as - the divergence of local model after training on new data points will be more, if higher gradient descent updates are performed. This can be because of variance in new data, use of better data points for local training rounds or greater training size. In case of divergence being relatively smaller, one can infer about participant using noisy or unrelated data points, which may not be acceptable for global federated model. Our framework would thus point at a possible way to fix a key limitation of the Shapley value framework (which we hope to build the link with in a future paper) - the fact that it only provides valuations for points within a fixed data set, and does not account for statistical aspects of the data and does not give a way to reason about points outside the data set.

V. EXPERIMENTS

In the paper, the experiments to validate the hypothesis of using blockchain smart contracts and federated contribution has been carried out with setting up a $(5+1=)6$ ethereum blockchain permissioned nodes setup in AWS cloud. Within this network, one of the node acts as federated aggregation server, and other three nodes act as federated clients. It is a consortium blockchain setup, with each individual client nodes own set of data. Figure 1 shows the architecture representation of the proposed experimental setup. The federated server node runs a python daemon process which listens to the events generated by smart contracts on the network. Upon receipt of events and based on event types, it either sends the global model to every node as a broadcast, or computes the aggregated version of the global model with latest gradients of local training, from individual federated clients. On federated clients, a similar python application is being executed, which listens to events and sends the encrypted local model, as required. It also acts as a service, which serves (*exposing remote REST based endpoint, which can be consumed by any other program to predict, based on input data*) machine learning model, and also is responsible for re-training of new batch of data points. The applications, which are responsible for directly interacting with blockchain nodes are termed as "dApp", as depicted in figure 1, which essentially means decentralized applications.

Federated Aggregator server node deploys both s '***communication***' and '**contribution records**' smart contracts. . All of the federated client nodes leverage the address of these two public contracts as a communication channel both for publishing events and listening to any generated events as well. The interface of the smart contract has been discussed in further.

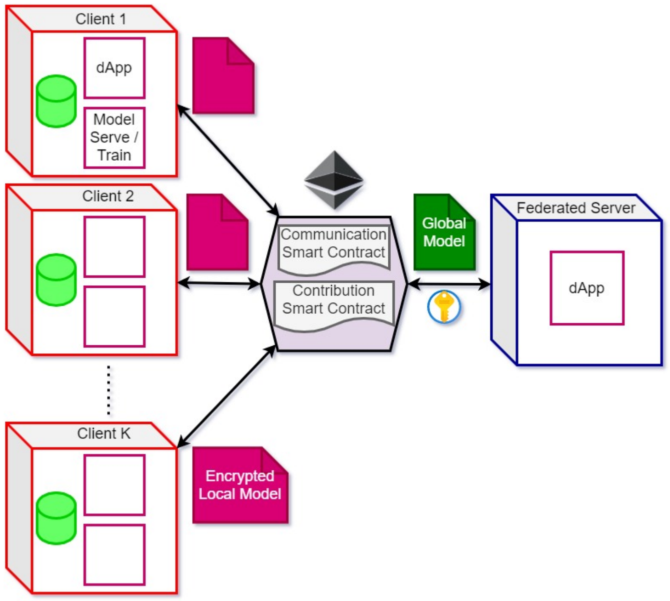

Fig. 1. In the above architecture diagram, individual nodes are running decentralized applications, with processes to serve and re-train data models. Two smart contracts, one for communication and one for contribution is deployed on the blockchain. Federated server is responsible to send the dynamically generated RSA key pair's public key, which would be used to encrypt the newly generated AES key of participant, used for model encryption. Encrypted models are published on the blockchain. Even though all the participants can receive the private encrypted model gradients, only the federated aggregation server can decipher and use the model, as it posses the RSA private key of respective public key.

A. Data description

In the paper, to test the hypothesis against standard data set, MNIST[33] and Fashion-MNIST[34] data sets have been used. In both the data sets, it has 10000 test data points, 60000 train data points and 10 target classes. We split the training data as 10000 for initial, "genesis" global model and remaining 50000 as training data for federated clients. For both the data sets, we have performed our experiments in two ways, by splitting our data sets described below -

1) Independent and Identical Data: The data is randomly split and distributed across all the federated clients, each having 10000 training data points with all the 10 target classes.
2) Non-Independent and Identical Data: The data is split and distributed across all the federated clients, each of the clients having training data with only 2 target class.

B. Blockchain smart contract interface

As discussed earlier, we have used two ethereum based smart contracts, one for communication and one for recording

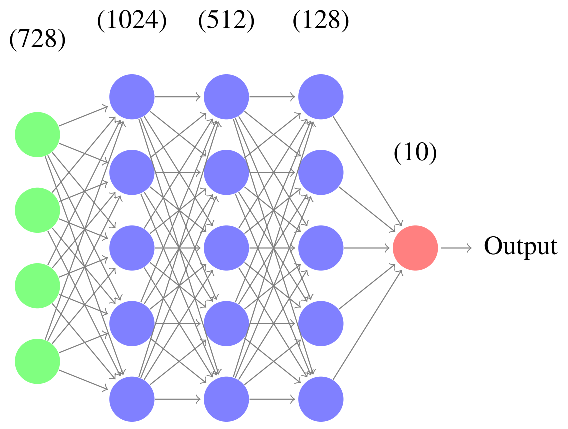

Fig. 2. In the architecture, the input layer contains 728 neurons, after flattening the image. Output layers contains 10 neurons, as the data set has 10 image labels or target classes. It contains 3 fully connected, hidden layers, each having 1024, 512 and 128 neurons respectively. The hidden layers used *relu*, and output layer used *softmax* as the activation functions.

contributions transparently on blockchain. Interface for both of the smart contracts have been defined as follows -

```
// Communication : Interface
pragma solidity >=0.8.0 <0.9.0;
contract Communication {
  event BCEvent(
    uint256 timestamp,
    bool is_encrypted,
    bytes event_type,
    bytes body
  );
  function publish(
    uint256 timestamp,
    bool is_encrypted,
    bytes memory event_type,
    bytes memory body
  ) public returns(uint ack) { }
}

// Contribution : Interface
pragma solidity >=0.8.0 <0.9.0;
contract Contribution {
  uint len = 5; //5 federated clients
  uint[] memory _clients = new uint[](5);
  function set_contribution(
    uint client_id,
    uint relative_contribution
  ) public returns(uint ack) {
  //only owner(federated server)
  //modifies state of _clients
  }
  function get_contributions()
  public view returns (uint memory) { } }
```

C. Neural network for image classification

Since the each image data in MNIST and Fashion-MNIST is 28x28 gray scale image, we have used artificial neural network[35] for image classification. Figure 2 describes the architecture of the neural network in depth. We have used **adam**[36] optimizer and **sparse categorical cross entropy**[37] as loss function, with 5 training epochs.

D. Results

Based on the data, blockchain network with smart contracts and neural network architecture described in earlier sections, we performed various experiments to test our hypothesis of computing federated contribution against various settings like higher contribution of a smaller group of participants, lower contributions by a subset of participants, noise in local data, while training by individual nodes. We have also calculated the overall accuracy of image classification of the aggregated model. In our experiments, we have changed the size of training samples, as it is the *variable* of the experiment. For each set of conditional environment for experiments, we have considered results for 4 different cases. The 4 cases are as *(i) MNIST data with non-IID split (ii) MNIST data with IID split (iii) Fashion-MNIST data with non-IID split and (iv) Fashion-MNIST data with IID split*. A well known and common observation across all the experimental setting is increase in machine learning model's accuracy performance, with increase number of training samples.

Figure 3 shows the performance of the aggregated machine learning model, from various clients. We do observe better and stable performance in case of data being split, across all the clients in *IID* manner. Figure 4 shows the federated contribution being increasing with increase in training data sample size. Also, since all the clients in this setting have contributed equally, their federated contribution values are quite close. One thing to notice, which is evident across all experiments (*can be observed in the graphs*) is, increase in federated contribution value, with increase in training size. This essentially validates our hypothesis of federated contribution being dependent on number of weight updates, which is directly proportional to higher training data (or training iterations). For distinct visuals, we have shown *client 3* in always green colour, in all the visualizations.

Figure 5 depicts the experiment, where *client 3* is under-performing by training on only 10% of what other participants are training. We can observe how the federated contribution value for *client 3* is very low, as compared to other participants. Figure 6 whereas shows the opposite, where *client 3* trains on 10 times more data points, puts in extra effort, logically, than other participants. This shows higher value in federated contribution for *client 3*, as relatively compared to others. Figure 7 is the final setting, where we added Gaussian noise, with noise value between 0 and 1, in training data of *client 3*, which essentially shows depletion in federated contribution of *client 3*, relative to others.

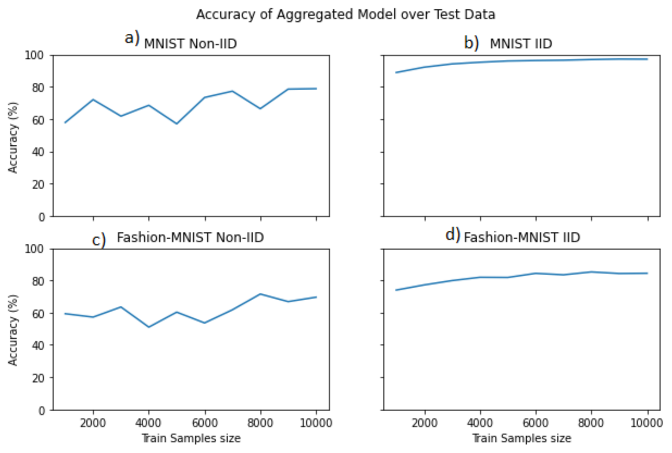

Fig. 3. (a) Accuracy of aggregated model in % vs training data sample size for MNIST data, split in Non-IID format (b) Accuracy of aggregated model in % vs training data sample size for MNIST data, split in IID format (c) Accuracy of aggregated model in % vs training data sample size for Fashion-MNIST data, split in Non-IID format (d) Accuracy of aggregated model in % vs training data sample size for Fashion-MNIST data, split in IID format

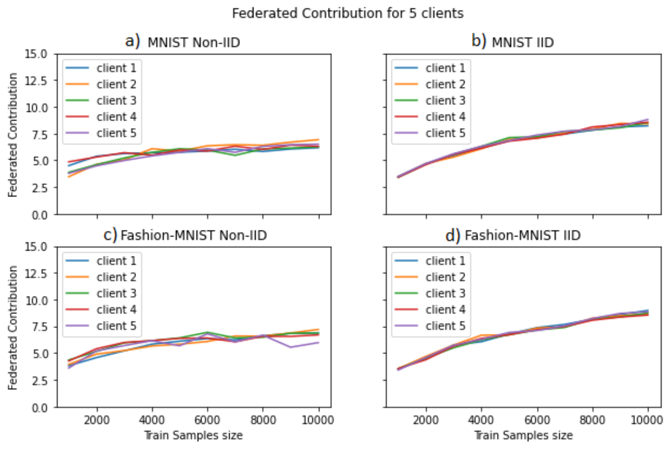

Fig. 4. (a) Federated contribution of equally contributing clients vs training data sample size for MNIST data, split in Non-IID format (b)Federated contribution of equally contributing clients vs training data sample size for MNIST data, split in IID format (c) Federated contribution of equally contributing clients vs training data sample size for Fashion-MNIST data, split in Non-IID format (d) Federated contribution of equally contributing clients vs training data sample size for Fashion-MNIST data, split in IID format

VI. Conclusion

In our paper, we studied our proposal of using smart contracts to establish a fair, transparent, secure and immutable incentivization mechanism in a consortium blockchain network for federated learning. We proposed a novel approach to calculate a unique scalar quantity, *federated contribution*, which quantifies the contribution of each participant in federated learning. Federated contribution is compatible with the machine learning algorithms which relies on weight parameters computed by gradient descents. We justified our proposed approach both empirically and theoretically. For future work in the given area, one can extend the federated contribution to non-gradient descents algorithms, or to heterogeneous federated learning. In the proposed method of calculating federated contribution, and using the relative federated contribution values for reward (or penalization) mechanism, we validated that it effectively penalizes under-performing participants, rewards over-performing participants and penalizing participants with noisy or malicious data points. This justifies our proposal of considering federated contribution as an adequate mechanism of quantifying participants' contribution in the consortium blockchain network. Future work will aim at building further the conceptual bridge between our weight-based contribution measure and Shapley values, under modified axioms that reflect the specificities of federated machine learning settings.

Acknowledgment

We would like to thank our colleagues Robert Otter, Nicolas X Zhang, Jiao Y Chang, Tulasi Das Movva, Thomas Eapen for reviews and discussions during the experimentation and

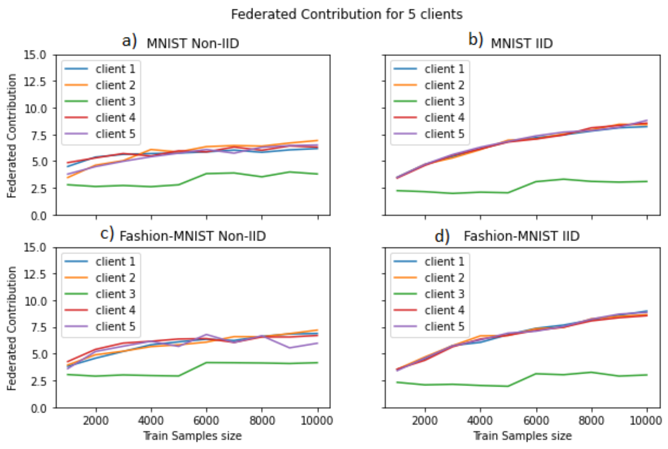

Fig. 5. (a) Federated contribution of highly contributing clients, but client 3 vs training data sample size for MNIST data, split in Non-IID format (b)Federated contribution of highly contributing clients, but client 3 vs training data sample size for MNIST data, split in IID format (c) Federated contribution of highly contributing clients, but client 3 vs training data sample size for Fashion-MNIST data, split in Non-IID format (d) Federated contribution of highly contributing clients, but client 3 vs training data sample size for Fashion-MNIST data, split in IID format

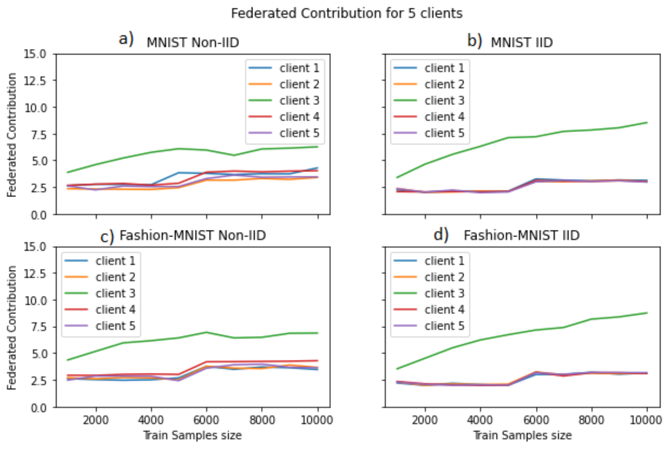

Fig. 6. Federated contribution of lowly contributing clients, but client 3 vs training data sample size for MNIST data, split in Non-IID format (b)Federated contribution of lowly contributing clients, but client 3 vs training data sample size for MNIST data, split in IID format (c) Federated contribution of lowly contributing clients, but client 3 vs training data sample size for Fashion-MNIST data, split in Non-IID format (d) Federated contribution of lowly contributing clients, but client 3 vs training data sample size for Fashion-MNIST data, split in IID format

evaluation. We would also like to thank J.P.Morgan Chase for supporting us in carrying out this effort.

References

[1] 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.

[2] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2009. [Online]. Available: http://www.bitcoin.org/bitcoin.pdf

[3] S. Fortunato, "Community detection in graphs," *Physics reports*, vol. 486, no. 3-5, pp. 75–174, 2010.

[4] O. Goldreich, S. Micali, and A. Wigderson, "Proofs that yield nothing but their validity or all languages in np have zero-knowledge proof systems," *Journal of the ACM (JACM)*, vol. 38, no. 3, pp. 690–728, 1991.

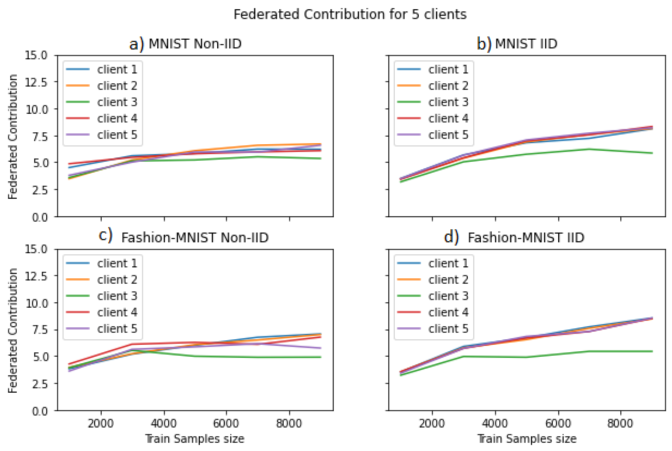

Fig. 7. (a) Federated contribution of highly contributing clients, but client 3 with noise vs training data sample size for MNIST data, split in Non-IID format (b)Federated contribution of highly contributing clients, but client 3 with noise vs training data sample size for MNIST data, split in IID format (c) Federated contribution of highly contributing clients, but client 3 with noise vs training data sample size for Fashion-MNIST data, split in Non-IID format (d) Federated contribution of highly contributing clients, but client 3 with noise vs training data sample size for Fashion-MNIST data, split in IID format

[5] B. Bünz, S. Agrawal, M. Zamani, and D. Boneh, "Zether: Towards privacy in a smart contract world," in *International Conference on Financial Cryptography and Data Security*. Springer, 2020, pp. 423–443.

[6] S. Bowe, A. Chiesa, M. Green, I. Miers, P. Mishra, and H. Wu, "Zexe: Enabling decentralized private computation," in *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2020, pp. 947–964.

[7] R. Cheng, F. Zhang, J. Kos, W. He, N. Hynes, N. Johnson, A. Juels, A. Miller, and D. Song, "Ekiden: A platform for confidentiality-preserving, trustworthy, and performant smart contracts," in *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2019, pp. 185–200.

[8] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *arXiv preprint arXiv:1812.06127*, 2018.

[9] V. Buterin, *Ethereum: A next-generation smart contract and decentralized application platform*, 2014. [Online]. Available: https://github.com/ethereum/wiki/wiki/White-Paper

[10] M. Swan, "Blockchain for business: Next-generation enterprise artificial intelligence systems," in *Advances in computers*. Elsevier, 2018, vol. 111, pp. 121–162.

[11] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.

[12] A. Bogner, M. Chanson, and A. Meeuw, "A decentralised sharing app running a smart contract on the ethereum blockchain," in *Proceedings of the 6th International Conference on the Internet of Things*, 2016, pp. 177–178.

[13] L. M. Bach, B. Mihaljevic, and M. Zagar, "Comparative analysis of blockchain consensus algorithms," in *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. IEEE, 2018, pp. 1545–1550.

[14] Ethereum and Tutorials, "Logging data from smart contracts with events," 2020. [Online]. Available: https://ethereum.org/en/developers/tutorials/logging-events-smart-contracts/

[15] Solidity and Documentation, "Events in contracts," 2016. [Online]. Available: https://docs.soliditylang.org/en/develop/contracts.html#events

[16] G. A. Pierro and H. Rocha, "The influence factors on ethereum transaction fees," in *2019 IEEE/ACM 2nd International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB)*. IEEE, 2019, pp. 24–31.

[17] S. Zhou, H. Huang, W. Chen, P. Zhou, Z. Zheng, and S. Guo, "Pirate: A blockchain-based secure framework of distributed machine learning in 5g networks," *IEEE Network*, vol. 34, no. 6, pp. 84–91, 2020.

[18] Y. J. Kim and C. S. Hong, "Blockchain-based node-aware dynamic weighting methods for improving federated learning performance," in *2019 20th Asia-Pacific Network Operations and Management Symposium (APNOMS)*. IEEE, 2019, pp. 1–4.

[19] U. Majeed and C. S. Hong, "Flchain: Federated learning via mec-enabled blockchain network," in *2019 20th Asia-Pacific Network Operations and Management Symposium (APNOMS)*. IEEE, 2019, pp. 1–4.

[20] 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)*. IEEE, 2019, pp. 151–159.

[21] Y. Li, C. Chen, N. Liu, H. Huang, Z. Zheng, and Q. Yan, "A blockchain-based decentralized federated learning framework with committee consensus," *IEEE Network*, 2020.

[22] X. Chen, J. Ji, C. Luo, W. Liao, and P. Li, "When machine learning meets blockchain: A decentralized, privacy-preserving and secure design," in *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 2018, pp. 1178–1187.

[23] B. A. Forouzan and D. Mukhopadhyay, *Cryptography and network security*. Mc Graw Hill Education (India) Private Limited, 2015.

[24] G. Singh, "A study of encryption algorithms (rsa, des, 3des and aes) for information security," *International Journal of Computer Applications*, vol. 67, no. 19, 2013.

[25] G. Wang, C. X. Dang, and Z. Zhou, "Measure contribution of participants in federated learning," in *2019 IEEE International Conference on Big Data (Big Data)*, 2019, pp. 2597–2604.

[26] M. Sundararajan and A. Najmi, "The many shapley values for model explanation," in *International Conference on Machine Learning*. PMLR, 2020, pp. 9269–9278.

[27] P. L. Ventre, M. M. Tajiki, S. Salsano, and C. Filsfils, "Sdn architecture and southbound apis for ipv6 segment routing enabled wide area networks," *IEEE Transactions on Network and Service Management*, vol. 15, no. 4, pp. 1378–1392, 2018.

[28] N. Khanezaei and Z. M. Hanapi, "A framework based on rsa and aes encryption algorithms for cloud computing services," in *2014 IEEE Conference on Systems, Process and Control (ICSPC 2014)*. IEEE, 2014, pp. 58–62.

[29] J. O'Hara, "Toward a commodity enterprise middleware: Can amqp enable a new era in messaging middleware? a look inside standards-based messaging with amqp," *Queue*, vol. 5, no. 4, pp. 48–55, 2007.

[30] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečnỳ, S. Mazzocchi, H. B. McMahan, *et al.*, "Towards federated learning at scale: System design," *arXiv preprint arXiv:1902.01046*, 2019.

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

[32] I. S. Dhillon and S. Sra, "Generalized nonnegative matrix approximations with bregman divergences," in *NIPS*, vol. 18. Citeseer, 2005.

[33] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010. [Online]. Available: http://yann.lecun.com/exdb/mnist/

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

[35] L. Deng, G. Hinton, and B. Kingsbury, "New types of deep neural network learning for speech recognition and related applications: An overview," in *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE, 2013, pp. 8599–8603.

[36] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[37] C. Louizos, M. Welling, and D. P. Kingma, "Learning sparse neural networks through l_0 regularization," *arXiv preprint arXiv:1712.01312*, 2017.