A Scalable Blockchain Approach for Trusted Computation and Verifiable Simulation in Multi-Party Collaborations

Ravi Kiran Raman,*† Roman Vaculin,† Michael Hind,† Sekou L. Remy,† Eleftheria K. Pissadaki,†
Nelson Kibichii Bore,† Roozbeh Daneshvar,† Biplav Srivastava,† and Kush R. Varshney†
*University of Illinois at Urbana-Champaign
†IBM Research

***Abstract*—In high-stakes multi-party policy making based on machine learning and simulation models involving independent computing agents, a notion of trust in results is critical in facilitating transparency, accountability, and collaboration. Using a novel combination of distributed validation of atomic computation blocks and a blockchain-based immutable audit mechanism, this work proposes a framework for distributed trust in computations. In particular we address the scalability problem by reducing the storage and communication costs using a lossy compression scheme. This framework guarantees not only verifiability of final results, but also the validity of local computations, and its cost-benefit tradeoffs are studied using a synthetic example of training a neural network.**

I. Introduction

If you were to receive a trained machine learning model or simulation output that was purportedly created using computationally-expensive open source code based on data that was also open, would you trust it? If the peer you received it from was known to cut corners sometimes because of resource constraints, would you trust it? In the absence of fully re-running the computation yourself, which could be prohibitive, you might not. And without such trust, collaboration would be out of the question.

Machine learning, data science, and large-scale computation have created an era of computation-driven inference, applications, and policymaking [1], which often have far-reaching consequences. Multi-agent sociotechnical systems that are tasked with working collaboratively on such tasks function by interactively sharing data, models, and results of local computation. However, when such agents are independent, they might not trust the validity of reported computations of other agents. Quite often, these computations are also expensive and time consuming, and thus infeasible for recomputation by the doubting peer as a general course of action. In such systems, creating an environment of trust, accountability, and transparency in the local computations of individual agents promotes collaborative operation. We propose such a methodology in this paper.

In AI, increasingly complex machine learning models with ever increasing complexity are being learned on vast datasets. Lack of access to the training process and the hyperparameters used therein do not only hinder scientific reproducibility, but also remove any basis to trust the reported results. In particular, applications interested in the trained model might not trust the agent performing the training. Other than sanity checks such as validation datasets, there exists no simpler way to guarantee correctness than complete retraining. This is impractical considering the extensive amounts of time and hardware required. It is thus important to create a pipeline for validated information-sharing that can not only be trusted by other agents, but also provide audits of the process that enable future development.

Let us also consider the case of policy design for disease control. OpenMalaria [2] is an open source simulation environment to study malaria epidemiology and the efficacy of control mechanisms, and is used extensively to design policies in tropical countries. Various health ministries and non-governmental organizations propose hypotheses regarding the disease and interventions, and study them by extensive simulations on the platform [3], [4]. In practice, trusted adoption of such results has been lacking because it relies on unquantified notions of reputation [5].

Calls have long been made for accountability and transparency in multi-agent computational systems [6]. Problems of enabling provenance in decision-making systems using an audit mechanism [7] and distributed learning in a federated setting with security and privacy limitations [8] have been considered recently. Reproducing results from research papers in AI has been found to be challenging as a significant fraction of hyperparameters and model considerations are not documented [9], [10].

However, there exists significant disparity and inconsistency in current information-sharing mechanisms that not only hinder access, but also lead to questionable informational integrity [11]. A framework for decision provenance enables

978-1-7281-1328-9/19/$31.00 ©2019 IEEE

accountability and trust by allowing transparent computational trajectories, and a unified, trusted platform for information sharing. Additionally, considering the cost of computation, it is also critical that any such trust guarantees are reliably transferable across agents.

A variety of notions of trust have been considered in multi-agent systems [12], [13]. At a fundamental level, trust is the belief that another party will execute a set of agreed-upon actions and report accurate estimates of the outcome [14], [15]. Computational trust translates to guaranteeing correctness of *individual steps* of the computation and integrity of the overall process, and can be decomposed into two elements:

- **Validation:** Individual steps of the computations are guaranteed and accepted to be correct.
- **Verification:** The integrity of the overall computation can be easily checked by other agents in the system.

Cryptographic techniques such as probabilistically checkable proofs [16] and zero knowledge proofs [17] have been studied to verify computations. In such methods, the results generated by an agent are verified using a cryptographic transcript of the process and avoid recomputation [18], [19]. However, practical implementations of such systems either require explicit knowledge of computational models, or restrict the class of verifiable functions considerably, beside significantly increasing computational cost for the computing agent who generates the transcript. In addition, such methods also require considerable amount of communication of proof transcripts by the client to each interested verifier. In this work, we instead adopt a distributed subset recomputation-based trust model focused on validation and verification using blockchain systems; this is the main contribution of this work.

Blockchain is a distributed ledger technology that enables multiple distributed, untrusting agents to transact and share data in a secure, verifiable and trusted manner through mechanisms providing transparency, distributed validation, and cryptographic immutability [20], [21]. A variety of applications that involve interactions among untrusting agents have benefited from the trust enabled by blockchains [22], [23]. More recently, blockchains have been used in creating secure, distributed, data sharing pipelines in healthcare [24] and genomics [25]. As such, blockchain is the perfect tool for establishing the type of trust for complex, long running computations of interest, and we describe its use in recording, sharing, and validating frequent audits (model parameters with the intermediate results) of individual steps of a computation. We describe how blockchain-based distributed validation and shared, immutable storage can be used and extended to enable efficient trusted verifiable computations.

A common challenge arising in blockchain-based solutions is its scalability [26]–[28]. The technology calls for significant peer-to-peer interaction and local storage of large records that include all the data generated in the network. These fundamental requirements result in significant communication and storage costs respectively. Thus, using the technology for large-scale computational processes over a large multi-agent networks is prohibitively expensive.

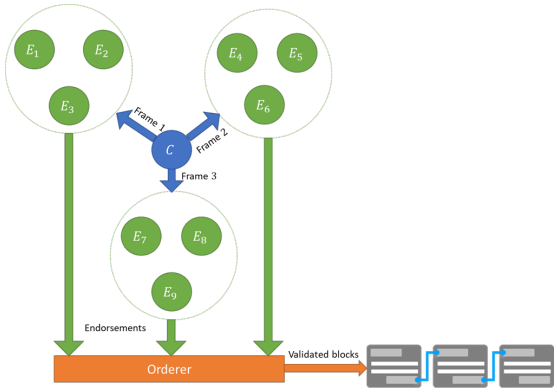

Fig. 1. Functional categorization of network: Clients run iterative algorithm; multiple frames validated in parallel by endorsers; orderers check consistency and append valid frames to blockchain.

In this paper, we address this challenge by developing a novel compression schema and a distributed, parallelizable validation mechanism that is particularly suitable for such large-scale computation contexts. We consider permissioned blockchains that do not require the proof of work mining process, and so do not require the impractical amounts of energy consumed by bitcoin-like blockchains [29]. The permissioned blockchains also enable fair task assignment and validation as well as the endorser sets can be assigned based on identity. In particular, we build this work and the experiments with respect to the IBM Hyperledger architecure. Further considerations need to be made regarding fairness and energy consumption for mining in public blockchains and is out of the scope of this paper.

II. Computation and Trust Model

Let us now mathematically formalize the computation model, and validation and verification requirements.

Consider the process that updates a system state, $X_t \in \mathcal{X} \subseteq \mathbb{R}^d$, over iterations $t \in \{1, 2, \ldots\}$, depending on the current state and an external randomness $\theta_t \in \Theta \subseteq \mathbb{R}^{d'}$, according to an atomic operation $f : \mathcal{X} \times \Theta \to \mathcal{X}$ as $X_{t+1} = f(X_t, \theta_t)$. We assume that $f(\cdot)$ is known to all agents, and is L-Lipschitz continuous in the system state, under the Euclidean norm, for all $\theta \in \Theta$ i.e.,

$$\|f(X_1, \theta) - f(X_2, \theta)\| \leq L \|X_1 - X_2\| . \tag{1}$$

That is, minor modifications to the state result in correspondingly bounded variation in the outputs. This is expected in most practical computational systems including simulations of physical or biological processes such as OpenMalaria.

Remark 1: Whereas Lipschitz continuity helps obtain strong theoretical guarantees on the cost-trust tradeoffs, the adaptive compression scheme introduced next can also be used in practical systems that lack the property.

We consider the multi-agent system shown in Fig. 1, where one agent, referred to as the *computing client*, runs the iterative algorithm and reports the states to other agents, called *peers*. Validation of intermediate states is performed by independent

subsets of peers referred to as *endorsers* through iterative recomputation of reported states from the the most recent validated state using $f(\cdot)$. The process of validation is referred to as an *endorsement* of the state. A reported state, $\tilde{X}_t$ is *valid* if it is within a margin, Δ_{val}, of the average state, $\hat{X}_t$, recomputed by the set of endorsers $\mathcal{E}_t$:

$$\left\|\tilde{X}_t - \hat{X}_t\right\| = \|\tilde{X}_t - \frac{1}{|\mathcal{E}_t|}\sum_{e\in\mathcal{E}_t} f(\tilde{X}_{t-1}, \theta_{t,e})\| \leq \Delta_{\text{val}}. \quad (2)$$

Validated frames are sent to orderers who append blocks to a blockchain ledger, maintaining an audit of validated states.

Verification is to ensure computational integrity through recorded audits of validated states. If the audits record states $\left\{\tilde{Y}_1, \tilde{Y}_2, \ldots\right\}$, then verification guarantees that for any t,

$$\mathbb{P}\left[\left\|\tilde{Y}_t - \mathbb{E}\left[f(\tilde{Y}_{t-1}, \theta)\right]\right\| \geq \Delta_{\text{ver}}\right] \leq \rho, \quad (3)$$

for some desired $\rho \in (0, 1)$, without explicit recomputation. Without loss of generality, validation requirements are stricter than for verification, i.e., $\Delta_{\text{val}} \leq \Delta_{\text{ver}}$. We now construct the system to address these two trust requirements.

III. Multi-agent Blockchain Framework

We now elaborate the system design, shown in Fig. 2, describing the role of the different agents. In particular, we elaborate the compression scheme used to reduce communication and storage costs. (All proofs are in the appendix.)

A. Client Operations

Clients compute the states iteratively, construct frames of iterates, compress, and report them to endorsers. We assume there exists an endorser assignment policy.

Owing to the Lipschitz continuity, $\|X_{t+1} - X_t\| \leq L\|X_t - X_{t-1}\|$. Thus state updates (differences) across iterates are bounded to within a factor of the deviation in the previous iteration. This can be leveraged to compress state updates using delta encoding [30], where states are represented in the form of differences from the previous state. Then, it suffices to store the state at certain checkpoints of the computational process, with the iterates between checkpoints represented by the updates.

We describe the construction inductively, starting with the initial state X_0, assumed to be the first checkpoint. Let us assume that the state reported at time t is $\tilde{X}_t$ and the computed state is X_t. Then, if $X_{t+1} = f(X_t)$, define $\Delta X_{t+1} = X_{t+1} - \tilde{X}_t$. To reduce the costs of communication and storage, we perform lossy compression (vector quantization) of the updates. Let the quantizer be represented by $Q(\cdot)$ and let the maximum quantization error magnitude be ϵ, i.e., if the client reports $\tilde{\Delta}X_t = Q(\Delta X_t)$, then, $\left\|\tilde{\Delta}X_t - \Delta X_t\right\| \leq \epsilon$.

Additionally, checkpoints are compressed using a Lempel-Ziv-like dictionary-based lossy compressor. Here, a dictionary of unique checkpoints is maintained and for each new checkpoint, if its state is within ϵ of an entry in the dictionary, its index is reported. If not, the state is added to the dictionary and its index is reported. Other universal vector quantizers can also be utilized for compressing checkpoints, and let us denote this quantizer by $\tilde{Q}(\cdot)$.

Let Δ_{quant} be the maximum magnitude of a state update within a frame, i.e., if $\|\Delta X_t\| > \Delta_{\text{quant}}$, the client creates a checkpoint at $t+1$, i.e.,

$$\tilde{X}_{t+1} = \begin{cases} \tilde{Q}(X_{t+1}) & \text{, if } t+1 \text{ is a checkpoint} \\ \tilde{X}_t + \tilde{\Delta}X_{t+1} & \text{, otherwise} \end{cases}. \quad (4)$$

The design parameters, $\epsilon, \Delta_{\text{quant}}$, are chosen such that the reports are accurate enough for validation.

Theorem 1: If $f(\cdot)$ is L-Lipschitz continuous and deterministic, and $\epsilon \leq \frac{\Delta_{\text{val}}}{L+1}$, then, a state $\tilde{X}_t$ is invalidated by an honest endorser only if there is a computational error of magnitude at least ϵ, i.e., $\left\|\tilde{X}_t - X_t\right\| \geq \epsilon$.

Corollary 1: $\tilde{X}_t$ is invalidated if $\left\|\tilde{X}_t - X_t\right\| \geq \Delta_{\text{val}} + L\epsilon$.

The necessary and sufficient conditions for invalidation in Thm. 1 and Cor. 1 imply that all errors above the tolerance are detected when the approximation error is made arbitrarily small. A variety of compressors, satisfying Thm. 1 can be used for lossy delta encoding—example, lattice vector quantizers [31].

Theorem 2: Let $\mathcal{B}(\Delta_{\text{quant}}) = \left\{x \in \mathbb{R}^d : \|x\| \leq \Delta_{\text{quant}}\right\}$ and let $\Delta X_t \sim \text{Unif}(\mathcal{B}(\Delta_{\text{quant}}))$, then the communication and storage cost per state update is $O\left(d \log\left(\frac{\Delta_{\text{quant}}}{\epsilon}\right)\right)$ bits.

This follows directly from the covering number of $\mathcal{B}(\Delta_{\text{quant}})$ using $\mathcal{B}(\epsilon)$ balls; other lattices incur similar costs.

Theorem 3: For any frame n, with checkpoint at T_n, maximum number of states in the frame, M_n, is bounded as

$$M_n \geq \min\left\{\frac{\log\left(\Delta_{\text{quant}} - \epsilon\right) - \log \delta_n}{\log L}, \bar{M}\right\}, \quad (5)$$

where $\delta_n = \|X_{T_n+1} - X_{T_n}\|$.

This is a sufficient condition for frame size, in terms of the magnitude of the first update in the frame. Naturally a small first iterate accommodates more states in the frame.

In practice, the Lipschitz constant L is unknown *a priori* and can be difficult to estimate. Underestimating L results in the use of a larger quantization error, resulting in invalidation of states owing to approximation errors. We draw insight from video compression, and propose the use of successive refinement coding [32] of the state updates. That is, a compression bit stream is generated for each state update such that the accuracy can be improved by sending additional bits from the bit stream. Successive refinement allows the endorsers to provide updates on the report such that the state accuracy can be iteratively improved.

Thus, if a state is invalidated, the client can either recompute the states or refine the reported state using successive refinement. Depending on the computation-communication cost tradeoff, the client chooses the more economical alternative. One efficient compression technique is lattice vector quantization with decreasing lattice sizes [33] to successively refine states. This reduces codebook size, and refinement communication cost if the same lattices are used. Using

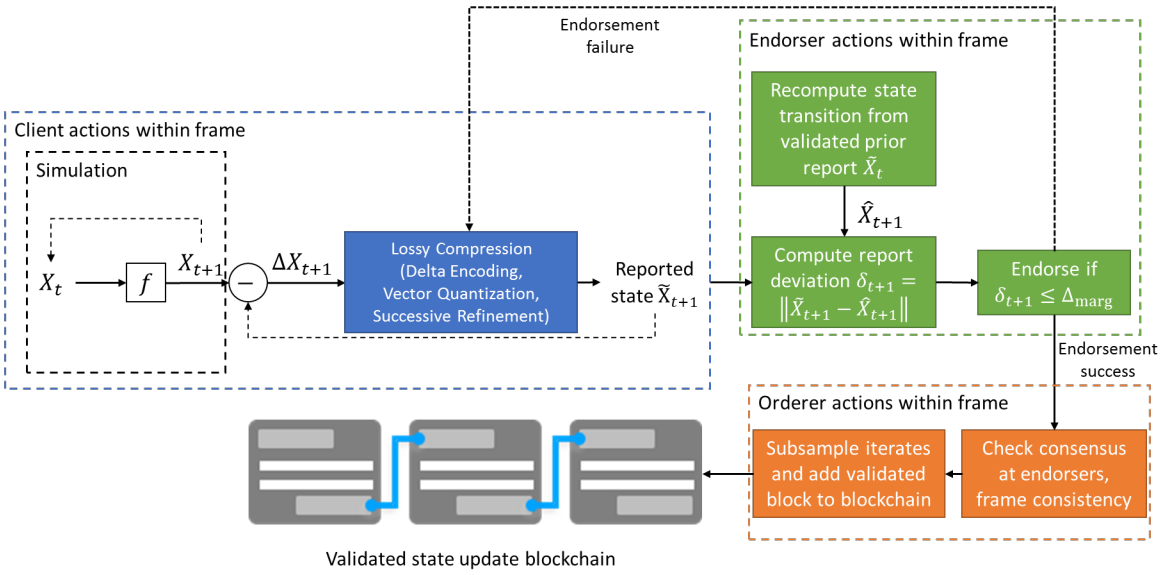

Fig. 2. Block diagram of computational trust policy.

successive refinement, clients can reduce approximation errors and recomputation frequency.

B. Endorser and Orderer Operations

We now define the role of endorsers in validating frames, as shown in Fig. 2. Here, we assume endorsers are honest and have equal latency, costs, and computational capacity.

Consider the set of endorsers $\mathcal{E}_n$ used to validate frame $\mathcal{F}_n$ of states. The endorsers decode the frame to obtain the set of reported states $\{\tilde{X}_t : t \in \mathcal{F}_n\}$. For each iterate $t \in \mathcal{F}_n$, each endorser $e \in \mathcal{E}_n$ recomputes the state $\hat{X}_t^{(e)} = f(\tilde{X}_{t-1}, \theta_t^{(e)})$. The endorsers evaluate the average state estimate, $\hat{X}_t = \frac{1}{\mathcal{E}_n} \sum_{e \in \mathcal{E}_n} \hat{X}_t^{(e)}$, and check the validation criterion (2). The frame is valid if all states are valid. The endorsements are reported to the orderer. Individual state validations can also be parallelized and finally verified for consistency, reducing the time for validation.

Theorem 4: Let $\epsilon < \frac{\Delta_{\text{val}}}{L+1}$. For a state at time t, if the average of m endorsers is used for validation,

$$\mathbb{P}\left[\left\|\tilde{X}_t - \hat{X}_t\right\| \ge \Delta_{\text{val}}\right] \le \frac{2d\tilde{\lambda}^2}{(\Delta_{\text{val}} - (L+1)\epsilon)^2}\left[1 + \frac{1}{m\tilde{\lambda}}\right]^2, \tag{6}$$

where $\tilde{\lambda}$ is the maximum eigenvalue of covariance matrix of of the quantized state vector.

Corollary 2: To guarantee validation with probability at least $1-\rho$, for a margin of deviation of Δ_{val}, where $\rho \le \frac{2d}{(\Delta_{\text{val}} - (L+1)\epsilon)^2}$, it suffices to use m endorsers, such that

$$m \ge \left[\left(\sqrt{\tfrac{\rho}{2d}}(\Delta_{\text{val}} - (L+1)\epsilon) - \tilde{\lambda}\right)\right]^{-1}. \tag{7}$$

Remark 2: From Thms. 1 and 4, and Cor. 1 and 2, we have the probabilistic guarantee that the computation steps are acceptable, catering to (3). Finally, from the data integrity guaranteed by the blockchain through the hash chain consistency, we know that adversarial computations, if any, are within an acceptable tolerance. To be precise, corruption of the ledger requires the collaboration of a majority of peers, and the alteration of all hash values to preserve consistency. This is considerably expensive over large networks, and practically infeasible for any adversarial agent.

Upon receiving frame endorsements, the orderer checks for consistency of checkpoints, adds a valid frame to the blockchain ledger if all prior frames have already been added, and broadcasts the block to other peers as depicted in Fig. 2. Since the state updates are stored on the immutable data structure of the blockchain, they enable verification of the computations by ensuring consistency of the hash chain. Since the states are pre-validated, it suffices to subsample the updates in a frame and store only a subset, i.e., one state is stored for every $K \approx \frac{\Delta_{\text{ver}}}{\Delta_{\text{val}}}$ iterates. The effective state update is the sum of the updates of the K intermediate iterates.

Thus a block stored on the ledger includes checkpoints and cumulative updates for K iterates, i.e., the audits $\tilde{Y}_\tau$ are

$$\tilde{Y}_{\tau+1} = \begin{cases} \tilde{Y}_\tau + \sum \tilde{\Delta} X_t & , \text{ if no checkpoint in } K \text{ iterates} \\ \tilde{X}_{t'} & , \text{ otherwise} \end{cases}, \tag{8}$$

where the sum is over the intermediate iterates, and t' is the next checkpoint. The audits are grouped into blocks and added to the blockchain ledger by the orderer.

Theorem 5: For sampling frequency $1/K$, Lipschitz constant L, deterministic $f(\cdot)$, and quantization error ϵ,

$$\left\|\hat{Y}_{\tau+1} - \tilde{Y}_{\tau+1}\right\| \le \left(L^K + 1\right) K\epsilon, \tag{9}$$

where $\hat{Y}_{\tau+1} = g(\tilde{Y}_\tau) := f^K(\tilde{Y}_\tau)$.
Thus, a viable subsampling frequency is to find the maximum K such that $(L^K + 1)K \le \frac{\Delta_{\text{ver}}}{\epsilon}$. Equivalent characterizations

for randomized computations can be obtained as well. The subsampling reduces the storage cost on the blockchain at the expense of recorded audit accuracy. Agents can also increase record accuracy over iterations, by dynamically adjusting the quantizers.

Remark 3: Here we have presumed that endorsers are honest. In practice, some endorsers could behave adversarially, and we account for this case by including sufficient redundancy in the endorser sets and using more robust estimates such as the median in place of the mean. That is, the endorser allocation is performed using coded computing such that the average number of adversaries in an allocation is less than half.

In practice, communications to some endorsers incur larger costs and latency, whereas some endorsers might be slow to recompute and validate. Such network costs and the presence of stragglers in the validation phase can be accounted for by encoding the redundancy into the validation task allocation using coded computing. Design of optimal codes for task allocation is however out of the scope of this paper and is to be considered in detail as part of future work.

C. Cost Analysis

First, let us consider the computational overhead involved in the multi-agent blockchain framework (MBF). If the expected number of endorsers per frame is E, then each state is computed $(1 + E)$-times as in the untrusted computation. Additional computation costs include that of compression and sampling, informally given by

$$C_{\text{batch}}^{(\text{comp.})} = (1+E)C_{\text{sim}}^{(\text{comp.})} + C_{\text{compression}}^{(\text{comp.})} + C_{\text{sampling}}^{(\text{comp.})}, \quad (10)$$

where $C_{\text{sim}}^{(\text{comp.})}, C_{\text{compression}}^{(\text{comp.})}, C_{\text{sampling}}^{(\text{comp.})}$ are the computational cost of an iteration, compression and decompression, and subsampling respectively.

The communication overheads include the state reports and metadata used for validation and coordination respectively. Assuming a bounded set of states, $\mathcal{X}$, such that $\max_{X\in\mathcal{X}} \|X\| = B \gg \Delta_{\text{quant}}$, the communication cost using vector quantization for $\bar{M}$ iterations is

$$C_{\text{batch}}^{(\text{comm.})} = O\left(\bar{M} d \log_2 \tfrac{\Delta_{\text{quant}}}{\epsilon} + d \log_2 \tfrac{B}{\epsilon} + C_{\text{meta}}^{(\text{comm.})}\right), \quad (11)$$

where $C_{\text{meta}}^{(\text{comm.})}$ is the metadata communication cost. Similarly, the storage cost for a subsampling frequency ν is

$$C_{\text{batch}}^{(\text{stor.})} = O\left(\nu \bar{M} d \log_2 \tfrac{\Delta_{\text{quant}}}{\epsilon} + d \log_2 \tfrac{B}{\epsilon} + C_{\text{meta}}^{(\text{stor.})}\right). \quad (12)$$

Thus MBF reduces communication and storage overheads at the expense of added computational cost. Through a careful analysis of the tradeoffs, we can adopt optimal compression schemes and subsampling mechanisms.

D. Example Application

Let us now elaborate the design for a synthetic example. Consider a client who trains a neural network using mini batch stochastic gradient descent (SGD) for classification on the MNIST dataset. However, the client has limited computational resources for training, and also does not share the private data used for training. Thus the client uses small batches to get faster gradient estimates. Peers do not trust client computations, not just because of its proclivity toward errors owing to computational limitations, but also possible malicious intent. We assume the peers also have access to much smaller set of private samples from the MNIST dataset, insufficient to train the network on their own. We then establish distributed trust using MBF as follows:

1) The client sets up the training with parametric inputs (network architecture, learning rates, batch sizes etc.) and shares them over the blockchain with other peers.
2) Client runs mini-batch SGD, compresses state updates (network weights), and reports frames to endorsers.
3) Endorsers rerun steps of mini batch SGD from validated states, compute deviation in network weights, endorse according to (2), and report endorsements to the orderer.
4) Orderer checks for consensus, subsamples iterates, constructs blocks, and adds them sequentially to blockchain.
5) Client reports validated, trained network to peers.

Since the experiment is run on the MBF platform, peers are assured of validity of steps of the training, and also have access to the blockchain to verify the computations.

IV. Experiments

In this section, we run some simple synthetic experiments using the MNIST database, for the scenario in Sec. III-D to understand MBF, the costs involved, and its benefits of enforcement. These experiments were selected to evaluate the efficacy of our approach over a domain that is familiar and common in the research community.

Let us consider a simple 3-layer neural network with 25 neurons in the hidden layer. Consider a client training the NN using mini-batch SGD using a small batch size of 10 samples per iteration and 1000 iterations, owing to resource limitations. The average precision of such a NN trained with gradient descent is 97.4%. We now wish to establish trust in the training process of the client.[1]

Since the training uses stochastic gradients, exact recomputation of the iterates is infeasible. Hence, we compare deviations from the average across sets of $m = 5$ endorsers for validation. We evaluate the computation and communication cost as a function of the validation tolerance. Since the NN converges to a local minimum according to SGD, we use decreasing tolerances across iterations, $\Delta_{\text{val}}(t) = \frac{\Delta_{\max}}{\log(t+1)}$.

We consider three main cases of the simulation:

1) **Base case:** Compression error less than tolerance, i.e., $\epsilon \leq \Delta_{\max}$; maximum frame size is 10% of no. of iterations.
2) **Coarse compression:** Large compression error, i.e., $\epsilon \geq \Delta_{\max}$ for at least some instances, and same base $\bar{M}$.
3) **Large frames:** Same base compression error, and maximum frame size is 20% of total number of iterations.

[1] Whereas this configuration is far from state of the art, it helps understand the trust environment better owing to its suboptimality.

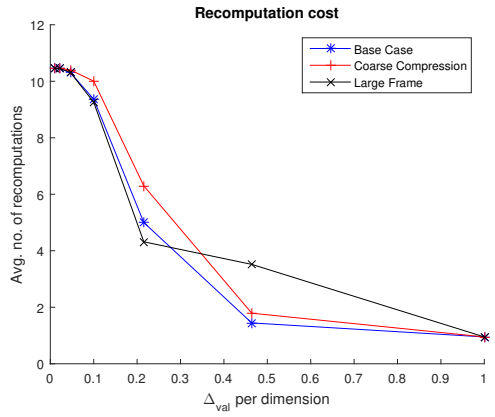

Fig. 3. Avg. recomputation cost tradeoff: Plot of avg. no. of gradient recomputations per iteration vs validation tolerance.

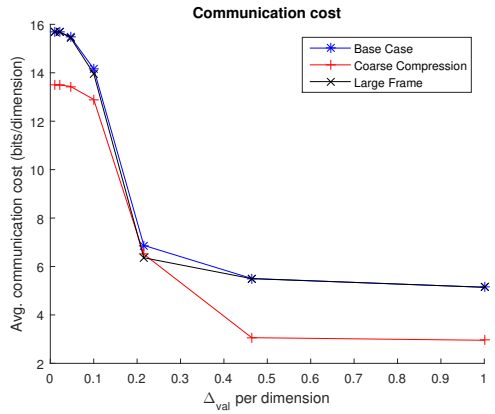

Fig. 4. Avg. communication cost tradeoff: Plot of avg. no. of bits per dimension communicated by clients vs validation tolerance.

In the base case, invalidation from approximation errors are more frequent in later iterations when the tolerance is also lower. However, with increasing iterations, the network weights are also closer to the minima. Thus approximation errors can be eliminated by successive refinement, as gradients estimates by the client also get more accurate. The presence of outliers and smaller batch sizes impact the initial iterations much more, which are reported with comparatively better accuracy, as required by the weaker validation criterion, therein only invalidating computational errors.

In comparison, in the case of coarse compression, approximation errors of the gradients are much more likely, resulting in more instances of invalidation. This translates to a higher number of gradient recomputations at the expense of reduced communication overhead on the compressed state updates. On the other hand, in the case of the extremely large frames, the endorsers validate longer sequences of states at once. Thus, each invalidation results in a recomputation of all succeeding states, therein increasing the number of recomputations from the base case. This case however reduces the number of frames and checkpoints, thereby reducing the average communication cost in comparison to the base case.

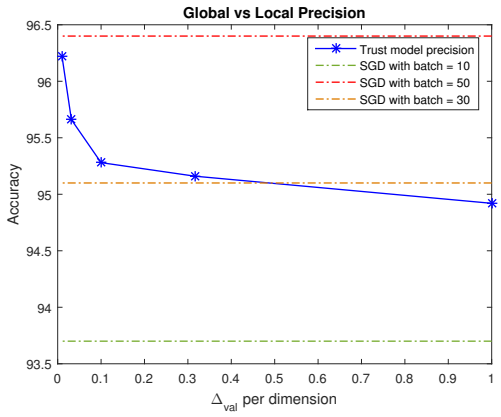

Fig. 5. Precision vs Trust: Plot depicts precision of trained NN satisfying (2). Eliminating spurious gradients through validation enhances training process.

In Fig. 3, the average number of gradient recomputations per iteration is shown for these three cases. As expected, this decays sharply as we increase the tolerance. Note that at either extreme, the three cases converge in the number of recomputations. This is owing to the fact that at one end all gradients are accepted whereas at the stricter end, most gradients are rejected with high probability, irrespective of the compression parameters. In the moderate tolerance range, we observe the tradeoffs as explained above. The corresponding communication cost tradeoff is shown in Fig. 4.

Fig. 5 shows the precision of the neural network trained under the validation requirement as compared against the networks trained with standard mini-batch SGD of batch sizes $10, 30,$ and 50. We note that the network trained with trust outperforms the case of vanilla SGD with the same batch size as it eliminates spurious gradients at validation. Increasing trust requirements (decreasing tolerance) results in improved precision of the model. In particular, it is worth noting that the strictest validation criterion results in performance that is almost as good as training with a batch size of 50. This is understandable as the endorsers validate only those gradients that are close to that of the case with mini batch of size 50. In fact, even when the trust requirements are relaxed, just eliminating outliers in gradients enhances training significantly.

These simple experiments not only highlight the role of trust in guaranteeing local and global consistency in the computational process, but also the cost tradeoffs involved in establishing them. Thus appropriate coding parameters can be chosen by studying the precision-cost, or other similar tradeoffs, implicit or explicit in the application.

V. Conclusion

In this paper we considered a multi-agent computational platform and the problem of establishing trust in computations performed by untrusting peers. Using a novel combination of blockchains and distributed consensus through recomputation, we assured validity of local computations and simple verification of computational trajectories. Using efficient compression

techniques, we also identified methods to reduce the communication and storage overheads concerned with establishing trust in such systems, thereby addressing the scalability challenge posed by blockchain systems.

Creation of such trusted platforms for distributed computation among untrusting agents allows for improved collaboration, and efficient data, model, and result sharing that is critical to establish efficient policy design mechanisms. Additionally they also result in creating unified platforms for sharing results ensuring scientific reproducibility.

REFERENCES

[1] D. J. Power, "Data science: supporting decision-making," *J. Decis. Sys.*, vol. 25, no. 4, pp. 345–356, Apr. 2016.

[2] T. Smith, N. Maire, A. Ross, M. Penny, N. Chitnis, A. Schapira, A. Studer, B. Genton, C. Lengeler, F. Tediosi, D. d. Savigny, and M. Tanner, "Towards a comprehensive simulation model of malaria epidemiology and control," *Parasitology*, vol. 135, no. 13, p. 15071516, Aug. 2008.

[3] J. D. Piette, S. L. Krein, D. Striplin, N. Marinec, R. D. Kerns, K. B. Farris, S. Singh, L. An, and A. A. Heapy, "Patient-centered pain care using artificial intelligence and mobile health tools: protocol for a randomized study funded by the us department of veterans affairs health services research and development program," *JMIR Res. Protocols*, vol. 5, no. 2, 2016.

[4] O. Bent, S. L. Remy, S. Roberts, and A. Walcott-Bryant, "Novel exploration techniques (NETs) for malaria policy interventions," in *Proc. AAAI Conf. Artif. Intell.*, Feb. 2018.

[5] S. L. Remy, O. Bent, and N. Bore, "Reshaping the use of digital tools to fight malaria," *arXiv:1805.05418 [cs.CY]*, May 2018.

[6] J. Nelson, "The operation of non-governmental organizations (ngos) in a world of corporate and other codes of conduct," *Corporate Social Responsibility Initiative*, Mar. 2007.

[7] J. Singh, J. Cobbe, and C. Norval, "Decision provenance: Capturing data flow for accountable systems," *arXiv:1804.05741 [cs.CY]*, Apr. 2018.

[8] D. Verma, S. Calo, and G. Cirincione, "Distributed AI and security issues in federated environments," in *Proc. Workshop Program 19th Int. Conf. Distrib. Comput. Netw.*, ser. Workshops ICDCN '18, Jan. 2018.

[9] O. E. Gundersen and S. Kjensmo, "State of the art: Reproducibility in artificial intelligence," in *Proc. 32nd AAAI Conf. Artif. Intell.*, New Orleans, USA, Feb. 2018.

[10] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," *arXiv:1709.06560v2 [cs.LG]*, Nov. 2017.

[11] W. G. V. Panhuis, P. Paul, C. Emerson, J. Grefenstette, R. Wilder, A. J. Herbst, D. Heymann, and D. S. Burke, "A systematic review of barriers to data sharing in public health," *BMC Public Health*, vol. 14, no. 1, p. 1144, Feb. 2014.

[12] R. Falcone, M. Singh, and Y.-H. Tan, *Trust in cyber-societies: integrating the human and artificial perspectives*. Springer Science & Business Media, 2001, vol. 2246.

[13] S. P. Marsh, "Formalising trust as a computational concept," Ph.D. dissertation, University of Stirling, 1994.

[14] P. Dasgupta, "Trust as a commodity," *Trust: Making and breaking cooperative relations*, vol. 4, pp. 49–72, 2000.

[15] S. D. Ramchurn, D. Huynh, and N. R. Jennings, "Trust in multi-agent systems," *Knowl. Eng. Review*, vol. 19, no. 1, 2004.

[16] M. Sudan, "Probabilistically checkable proofs," *Commun. ACM*, vol. 52, no. 3, pp. 76–84, Mar. 2009.

[17] O. Goldreich and Y. Oren, "Definitions and properties of zero-knowledge proof systems," *J. Cryptology*, vol. 7, no. 1, pp. 1–32, Dec. 1994.

[18] M. Walfish and A. J. Blumberg, "Verifying computations without reexecuting them," *Commun. ACM*, vol. 58, no. 2, Jan. 2015.

[19] B. Parno, J. M. McCune, and A. Perrig, *Bootstrapping trust in modern computers*. Springer Science & Business Media, 2011.

[20] K. Croman, C. Decker, I. Eyal, A. E. Gencer, A. Juels, A. Kosba, A. Miller, P. Saxena, E. Shi, E. G. Sirer, D. Song, and R. Wattenhofer, "On scaling decentralized blockchains," in *Financial Cryptography and Data Security*, ser. Lecture Notes in Computer Science, J. Clark, S. Meiklejohn, P. Y. A. Ryan, D. Wallach, M. Brenner, and K. Rohloff, Eds. Berlin: Springer, 2016, vol. 9604, pp. 106–125.

[21] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. D. Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, S. Muralidharan, C. Murthy, B. Nguyen, M. Sethi, G. Singh, K. Smith, A. Sorniotti, C. Stathakopoulou, M. Vukolić, S. W. Cocco, and J. Yellick, "Hyperledger fabric: A distributed operating system for permissioned blockchains," in *Proc. 13th EuroSys Conf.*, ser. EuroSys '18, Apr. 2018, pp. 30:1–30:15.

[22] D. Tapscott and A. Tapscott, *Blockchain Revolution: How the Technology behind Bitcoin is Changing Money, Business, and the World*. New York: Penguin, 2016.

[23] M. Iansiti and K. R. Lakhani, "The truth about blockchain," *Harvard Bus. Rev.*, vol. 95, no. 1, pp. 118–127, Jan. 2017.

[24] J. Tsai, "Transform blockchain into distributed parallel computing architecture for precision medicine," in *2018 IEEE 38th Int. Conf. Distrib. Comput. Systems (ICDCS)*, Jul. 2018.

[25] H. I. Ozercan, A. M. Ileri, E. Ayday, and C. Alkan, "Realizing the potential of blockchain technologies in genomics," *Genome Research*, 2018.

[26] Z. Koticha, "2018: Blockchain scaling > all else," https://medium.com/thunderofficial/2018-blockchain-scaling-all-else-7937b660c08, Jun. 2018.

[27] R. K. Raman and L. R. Varshney, "Distributed storage meets secret sharing on the blockchain," in *Proc. Inf. Theory Appl. Workshop*, Feb. 2018.

[28] ——, "Dynamic distributed storage for blockchains," in *Proc. IEEE Int. Symp. Inf. Theory*, Jun. 2018, pp. 2619–2623.

[29] M. Vukolić, "Rethinking permissioned blockchains," in *Proc. ACM Workshop Blockchain, Cryptocurrencies and Contracts*, ser. BCC '17, Apr. 2017, pp. 3–7.

[30] C. W. Granger and R. Joyeux, "An introduction to long-memory time series models and fractional differencing," *J. Time Ser. Anal.*, vol. 1, no. 1, pp. 15–29, Jan. 1980.

[31] S. D. Servetto, V. A. Vaishampayan, and N. J. A. Sloane, "Multiple description lattice vector quantization," in *Proc. IEEE Data Compression Conf. (DCC 1999)*, Mar. 1999, pp. 13–22.

[32] W. H. R. Equitz and T. M. Cover, "Successive refinement of information," *IEEE Trans. Inf. Theory*, vol. 37, no. 2, Mar. 1991.

[33] D. Mukherjee and S. K. Mitra, "Successive refinement lattice vector quantization," *IEEE Trans. Image Process.*, vol. 11, no. 12, pp. 1337–1348, Dec. 2002.

[34] A. W. Marshall and I. Olkin, "Multivariate chebyshev inequalities," *Ann. Math. Stat.*, vol. 31, no. 4, pp. 1001–1014, Dec. 1960.

APPENDIX

Let us assume that $\tilde{X}_t$ is a valid state. Then,

$$\begin{aligned}\left\|\hat{X}_{t+1}-\tilde{X}_{t+1}\right\| &\le \left\|\hat{X}_{t+1}-X_{t+1}\right\| + \left\|X_{t+1}-\tilde{X}_{t+1}\right\| \\ &\le L\left\|\tilde{X}_t - X_t\right\| + \left\|\tilde{\Delta}X_t - \Delta X_t\right\| \quad (13)\\ &\le L\left\|\tilde{\Delta}X_{t-1} - \Delta X_{t-1}\right\| + \epsilon \quad (14)\\ &\le (L+1)\epsilon \le \Delta_{\text{val}}, \quad (15)\end{aligned}$$

where (13) follows from Lipschitz continuity, (14) from the definition of the compressed state updates, and (15) from the quantization error bound.

Without loss of generality, let $n = 1, T_n = 0$. Then, $M_n = t$ implies that

$$\begin{aligned}\Delta_{\text{quant}} &\le \|\Delta X_t\| = \left\|X_{t+1} - \tilde{X}_t\right\| \\ &\le \|X_{t+1} - X_t\| + \left\|X_t - \tilde{X}_t\right\| \quad (16)\\ &\le L^t \|X_1 - X_0\| + \epsilon = L^t\delta_1 + \epsilon, \quad (17)\end{aligned}$$

where (16) follows from the triangle inequality, and (17) from the Lipschitz continuity and the quantization error. Thus the result follows.

For a d-dimensional random variable X with mean and covariance μ, V respectively, from the multidimensional Chebyshev inequality [34],

$$\mathbb{P}\left[\|X-\mu\|_{V^{-1}} > t\right] \leq \frac{d}{t^2}.$$

Then, using the fact that $\lambda_{min}\|x\| \leq \|x\|_A \leq \lambda_{max}\|x\|$, for any vector x and matrix A with minimum and maximum eigenvalues $\lambda_{min}, \lambda_{max}$, we have

$$\mathbb{P}\left[\|X-\mu\| > t\right] \leq \frac{\lambda^2 d}{t^2}, \tag{18}$$

where λ is the maximum eigenvalue of V.

Further, from the bound on the quantization error, we can observe that for $\tilde{X}_{t+1} = X_{t+1} + Z_t$, where $Z_t \in \mathcal{B}(\epsilon)$. Then,

$$\left\|\mathbb{E}\left[X_{t+1}\right] - \mathbb{E}\left[\tilde{X}_{t+1}\right]\right\| \leq \epsilon.$$

Then,

$$\begin{aligned}
\left\|\mathbb{E}\left[\tilde{X}_{t+1}\right] - \mathbb{E}\left[\hat{X}_{t+1}\right]\right\| &= \left\|\mathbb{E}\left[\tilde{X}_{t+1}\right] - \mathbb{E}\left[f(\tilde{X}_t,\theta)\right]\right\| \\
&\leq \left\|\mathbb{E}\left[X_{t+1}\right] - \mathbb{E}\left[\tilde{X}_{t+1}\right]\right\| + \left\|\mathbb{E}\left[f(X_t,\theta - f(\tilde{X}_t,\theta)\right]\right\| && (19)\\
&\leq \epsilon + L\left\|\mathbb{E}\left[X_t\right] - \mathbb{E}\left[\tilde{X}_t\right]\right\| && (20)\\
&\leq (L+1)\epsilon, && (21)
\end{aligned}$$

where (19) follows from the triangle inequality, and (20) follows from the bound on the quantization error, the fact that $\mathbb{E}\left[\|X\|\right] \geq \|\mathbb{E}\left[X\right]\|$, and the Lipschitz continuity.

Finally, for any $\alpha \in (0,1)$,

$$\begin{aligned}
&\mathbb{P}\left[\left\|\tilde{X}_{t+1} - \hat{X}_{t+1}\right\| \geq \Delta_{\text{val}}\right] \\
&\quad\leq \mathbb{P}\left[\left\|\tilde{X}_{t+1} - \mathbb{E}\left[\tilde{X}_{t+1}\right]\right\| \geq \alpha(\Delta_{\text{val}} - (L+1)\epsilon)\right] \\
&\quad+ \mathbb{P}\left[\left\|\hat{X}_{t+1} - \mathbb{E}\left[\hat{X}_{t+1}\right]\right\| \geq \bar{\alpha}(\Delta_{\text{val}} - (L+1)\epsilon)\right] && (22)\\
&\quad\leq \frac{d}{(\Delta_{\text{val}} - (L+1)\epsilon)^2}\left(\frac{\tilde{\lambda}^2}{\alpha^2} + \frac{1}{m^2(1-\alpha)^2}\right) && (23)\\
&\quad\leq \frac{2d\tilde{\lambda}^2}{(\Delta_{\text{val}} - (L+1)\epsilon)^2}\left(1 + \frac{1}{m\tilde{\lambda}}\right)^2, && (24)
\end{aligned}$$

where (22) follows from the triangle inequality and the union bound, and (21), and (23) from (18). Finally, (24) is obtained by maximizing the bound over $\alpha \in (0,1)$.

First, $g(\cdot)$ is L^K-Lipschitz continuous. Then,

$$\begin{aligned}
&\|\hat{Y}_{\tau+1} - \tilde{Y}_{\tau+1}\| && (25)\\
&\quad\leq \|g(\tilde{Y}_\tau) - g(Y_\tau)\| + \|Y_{\tau+1} - \tilde{Y}_{\tau+1}\| && (26)\\
&\quad\leq L^K\|\tilde{Y}_\tau - Y_\tau\| + \left\|\sum_{t\in\mathcal{T}_\tau}\left(\tilde{\Delta}X_t - \Delta X_t\right)\right\| && (27)\\
&\quad\leq (L^K+1)K\epsilon,
\end{aligned}$$

where (26) follows from the triangle inequality, and (27) follows from the Lipschitz inequality and (8).