Sustainable Spectrum Crowdsensing

Yijing Zeng*, Bangya Liu*, Yilong Li, *Student Member, IEEE,*
Domenico Giustiniano, *Senior Member, IEEE,* and Suman Banerjee, *Fellow, IEEE*

***Abstract*—Spectrum crowdsensing is a paradigm where participants upload their collected spectrum data to the cloud for extracting analytics. First movers like Microsoft Spectrum Observatory and Electrosense, though with support from leading industry, research, and government, still suffer from sustainability challenges. In this paper, we present Fiesta, a sustainable framework for spectrum crowdsensing. On the technology side, we use federated learning and blockchain to decentralize the data analysis computations. For individual participants, minimal invasion of privacy suppresses concerns regarding large-scale adoption. From organizations' perspectives, using blockchain avoids single point of failure and enhances the robustness of the entire system against malicious attacks. On the policy side, we propose a reward quantification mechanism to motivate engagement. Potential funding sources to ensure ongoing sustainability are also discussed. We have demonstrated Fiesta through simulation testbeds and real-world deployments with two demo tasks. Results show that Fiesta, as a decentralized framework, can preserve user privacy, enhance system robustness, maintain data fidelity compared with traditional methods, and fairly reward participants. We believe Fiesta is a stepping stone for the future spectrum crowdsensing paradigm.**

***Index Terms*—Communications technology, Communication systems, Radio communication, Radio spectrum management**

I. INTRODUCTION

The scarcity of wireless spectrum and the exploding mobile data traffic compel regulatory authorities to consider opening previously allocated bands for secondary users, such as TV whitespace [32] and CBRS band [27], in addition to leveraging existing unlicensed bands [7]. The feasibility of such endeavors depends strongly on a comprehensive understanding of how these bands are used by primary users. This requires large-scale spectrum sensing measurements across frequency, temporal, and spatial domains.

To fulfill this requirement, crowdsensing [29] is a promising spectrum measurement paradigm. For example, Microsoft Spectrum Observatory [1] and Electrosense [25] make efforts to achieve national-wide or even continental-wise spectrum measurement using Software Defined Radio (SDR) platforms.

However, those prior projects were designed without considering sustainability. Specifically, for Electrosense, the problem was that the sensors, although low-cost, were deployed by merely researchers and a few practitioners. Electrosense made a substantial effort to increase the interest of users, and thus, the deployment density [5]. Nevertheless, the need to acquire dedicated embedded hardware and the usability of the front-end interface limited its ability to engage lay users, whose participation is critical for achieving dense deployments. As for Microsoft Observatory, it is an earlier attempt, that required expensive Ettus board, that could be afforded only by researchers. The front-end did not adequately address user experience aspects. Most importantly, both projects are in lack of a mechanism to proactively encourage participation.

With previous lessons in mind, we envision that a large-scale spectrum crowdsensing system should have a three-tier architecture. At the bottom, mobile devices owned by individuals have spectrum sensing capability and collect spectrum measurements. In the middle, servers from organizations perform data analysis for spectrum-related applications. At the top, the administrator coordinates the whole system, for example deciding the target frequency range and what kind of spectrum-related data analysis should be performed.

Despite the proposed three-tier architecture, there are still several challenges before realizing the vision at scale.

- Prior efforts, including TxMiner [33] and BigSpec [30], require contributors to upload raw data records with metadata, e.g., GPS location and timestamps, to a centralized cloud for analysis. However, this information is private for participants, especially in the scenario of mobile sensing. The spectrum data itself could also be sensitive, and in some cases, information could be decoded from captured IQ data or PSD (Power Spectrum Density) data.
- Another technical challenge is that sending data to a centralized data analysis platform poses the risk of single point of failure and is more error-prone as one single entity owns the computing infrastructure entirely.
- On the policy side, without a proper reward mechanism to compensate, all these privacy concerns discourage mobile spectrum crowdsensing participants from contributing more data. Organizations may be unwilling to collaborate with little incentive.

To address these concerns, we present **Fiesta**, a sustainable framework for large-scale spectrum crowdsensing practicing the three-tier architecture.

First, we propose to move partial data analysis from the cloud server to mobile devices. We introduce and adapt federated learning [13] into the spectrum measurement context, where the user shares useful information in the format of a machine learning model instead of raw data. The scenario becomes challenging considering ordinary federated learning requires the participants to be synchronized in the sense that the model aggregation is performed after all of the participants have completed a local training round. However, synchronization is hard to achieve when scaling to a large

---

Y. Zeng, B. Liu, Y. Li, and S. Banerjee are with University of Wisconsin-Madison. E-mail: {yijingzeng, bangya, yilong, suman}@cs.wisc.edu. D. Giustiniano is with IMDEA Networks Institute. Email: domenico.giustiniano@imdea.org

A short version of this paper was presented in the IEEE DySPAN 2024 [31].

Manuscript received XXX, 2024; revised XXX, 2025; accepted XXX, 2025.

*These authors contributed equally to this work.

scale where each device has its schedule of sampling and training, so tackling this asynchronous setup is a must. We propose a novel aggregation policy and empirically show its enhancement compared with vanilla federated learning.

Second, we eliminate the risk of single point of failure and enhance system robustness by using a distributed ledger, i.e., blockchain, to aggregate and archive the learned model. We design a block structure that contains the current global model so that sharing the data analysis result is naturally supported. To make it more efficient and pragmatic for large-scale deployment, we also solve several implementation issues, including how to support oversized models and the evolution of the model structure.

Third, a reward quantification mechanism based on this blockchain is carefully designed to further promote the participation of both individuals and organizations. In the vision of Fiesta, individuals and organizations could gain rewards either by contributing useful information or joining the blockchain and helping with model aggregation. Administrators and authorities who could benefit from gathered spectrum data (e.g., FCC, 3GPP, operators, and research institutions) are expected to maintain this blockchain and provide rewards.

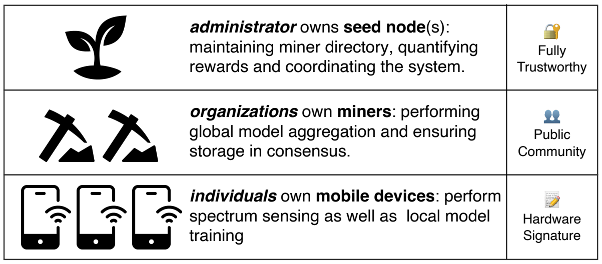
To demonstrate the efficacy of Fiesta, we have simulated, implemented, as well as deployed Fiesta using mobile devices connected to our spectrum sensor boards to learn the utilization of TV and LTE bands.

In summary, we make the following contributions:

- To the best of our knowledge, this is the first work that proposes and designs a sustainable spectrum crowdsensing system at scale. The three-tier architecture based on individuals who own mobile devices with sensing capabilities, data aggregation servers from multiple organizations, and the administrator coordinating the whole system is crucial for long-term operation.
- We solved multiple technical and policy-related challenges to realize this vision. By adapting federated learning and blockchain to the spectrum sensing context, we have mitigated the privacy concerns of the individual participants, addressed the risk of single point of failure, and encouraged larger engagement by designing a reward quantification mechanism based on the blockchain. Potential funding sources to ensure ongoing sustainability are also discussed.
- We have implemented and deployed Fiesta in a mid-size US city for 3 months and collected 220K lines of PSD data records for 600 hours using Android smartphones. Our model converges to 4dB error for TV channels' power regression across space and 5dB error for reconstructing LTE band PSD data. The dataset is available at https://www.kaggle.com/datasets/neutrinoliu/fiesta, and the code is available at https://fiesta4spectrum.github.io/.

II. SYSTEM DESIGN

In this section, we present the design and the key ideas of Fiesta. We first present the architecture of Fiesta, next explain how the learning process works in Fiesta, then discuss the reward mechanisms to encourage participants, followed by a subsection justifying the necessity of blockchain.



	administrator owns **seed node**(s): maintaining miner directory, quantifying rewards and coordinating the system.	Fully Trustworthy
	organizations own **miners**: performing global model aggregation and ensuring storage in consensus.	Public Community
	individuals own **mobile devices**: perform spectrum sensing as well as local model training	Hardware Signature

Fig. 1. Layered architecture of Fiesta and the trust model.

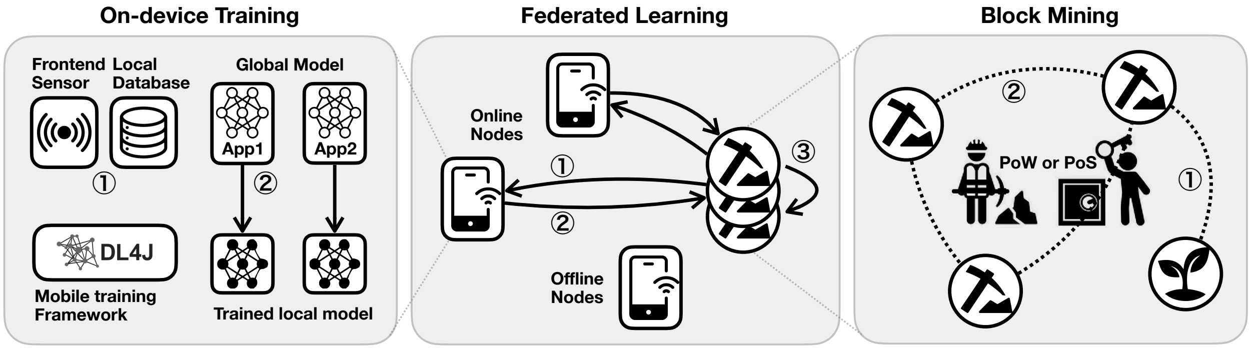
A. Architecture

There are three different roles in Fiesta, as shown in Fig. 1, individuals with mobile devices, organizations with miners, and the administrator with seed node(s).

In Fiesta, *individuals* own the **mobile devices**, such as smartphones, with spectrum sensing capabilities. The frontend of these mobile devices has the capability of sensing the spectrum at fine granularity over a large set of frequency bands. This sensing capability might probably be part of future generations of smartphones, depending on the application and the need of uploading IQ or PSD data. In our setup, the frontend consists of a small-factor SDR attached to the smartphone through USB. Mobile devices in Fiesta setup mainly operate in two modes, data collection mode and learning mode. When in the data collection mode, they record the spectrum readings and the corresponding meta-data, i.e. latitude, longitude, and timestamp. When in the learning mode, they compute the local updated model based on the current global model and the collected local spectrum data, then communicate with miners for model aggregation. In our trust model, radio frontend (i.e. hardware part) is generally regarded as trustworthy considering its tampering difficulty. Yet for the software, especially the machine learning procedure, we use digital certificates to enforce traceability.

Organizations own the **miners** in Fiesta. Organizations could be cellular operators implementing an extension of Minimization of Drive Tests (MDT) measurements, or over-the-top organizations like Microsoft and Electrosense, who are interested in spectrum data. Miners will gather a bundle of updates from mobile devices and batch them into a block, then calculate the aggregated global model. They also feed the up-to-date global model with mobile devices. All the versions of the global model and the corresponding updates from mobile devices to get the global model are recorded in the form of a blockchain. The Proof-of-Work (PoW) serves as the consensus protocol among the miners.

The *administrator* of the community (e.g. the government or the regulatory authority like the FCC, 3GPP) owns the **seed node(s)** in Fiesta. These seed node(s) have two specific functions for the blockchain. First, they maintain a directory of working miners such that new mobile devices and miners joining the community know where to contact the existing miners. Second, they periodically scan the blockchain to quantify rewards for each participant. The administrator, e.g. FCC and 3GPP, also designs the machine learning models for spectrum-related applications, specifies which particular spectrum and spatio-temporal area is of interest, and offers monetary rewards for participants.



① mobile device gathers data with SDR, save the data in the local database
② Train the fetched global model with local dataset, leveraging training framework

① Fetch latest global model from available miner node
② Upload the trained local model to available miner
③ Aggregate local updates into the next generation global model

① Seed node shares a genesis block to initiate a new sensing task
② Miners share local updates and mine new blocks containing the latest global model

Fig. 2. Three fundamental components of Fiesta: on-device training, federated learning, blockchain.

B. Learning process

Fig. 2 illustrates the model learning process in Fiesta, where red arrows indicate the cycle of federated learning and blue arrows indicate how a new block comes into birth.

Local training at mobile devices. The main idea is to use a learning model to describe the spectrum data in accordance to specific needs of the application. The mobile device starts training when the device is in charging as long as the user enables the learning mode in the app and will keep training whenever there are new global model downloadable from miner. For example, mobile device will train the pulled global model for one round (a fixed number of epochs) based on local PSD data (frequency information) and GPS readings (spatial information) to estimate TV channels' power at different locations, as shown in the left box of Fig. 2. Then, the mobile device uploads that updated model with other information including device identity, size of training set, and the training loss local training to any miner for performance monitoring and reward calculation. All this information is signed using its private key for identity verification. Note that the optimizer is reset to the same initial values after each round, so there is no need to worry about the learning rate varying significantly among the mobile devices.

Updates aggregation at miners. As shown in Fig. 2, miners receive updates, forward to neighbors, batch them into a block, and compute the new global model. We use a modified version of FedAvg [19] to deal with asynchronous model aggregation. FedAvg is the most popular model aggregation method in the synchronous setting. It computes the weighted average of the model updates with the weights equal to the number of data records, and it is guaranteed to converge with non-iid (independent and identically distributed) data for strongly convex and smooth problems [17], although slower than the iid case. Formally, suppose client $i \in \{1, \dots, C\}$ receives the same copy of the global model $\mathbf{w}_t$ at the beginning of round $t+1 (t \geq 0)$. Then, after local training, all clients (or only a fraction of all clients selected by the central server) generate their local update $\mathbf{w}^i_{t+1}$ using their local n_i number of data records. To get the new global model after round $t+1$, the central server in FedAvg computes:

$$\mathbf{w}_{t+1} = \sum_{i=1}^{C} \frac{n_i}{N} \mathbf{w}^i_{t+1}, \qquad (1)$$

where $N = \sum_{i=1}^{C} n_i$. Finally, this new global model $\mathbf{w}_{t+1}$ is pushed to all clients for the next round of training.

Challenge of Asynchrony. Compared with ordinary federated learning, Fiesta needs to handle the challenge of asynchronoy. Due to the heterogeneity of the capability of each mobile devices, the different amounts of data each mobile device collects, and the latency and throughput of the wireless connections during model learning, it is expected to encounter a variety of speeds per local round of training. Furthermore, due to the block propagation delay within the miners' network and the possible forking of the blockchain, mobile clients might also download different versions of the global model at the same time. The challenge of asynchrony is further exacerbated by the freedom of joining and exiting Fiesta.

EWMA enhanced FedAvg. In the context of synchronous federated learning, it is expected that $block_{t+2}$ contains those model updates derived from only the global model w_{t+1} outputted in $block_{t+1}$. However, due to above-mentioned asynchronous nature of Fiesta, $block_{t+2}$ may also contain model updates based on the global model w_t outputted in $block_t$, uploaded by straggling clients. We define these updates in $block_{t+2}$ but based on global model w_t as "1 block delayed". In reality, the updates may be several blocks delayed instead of merely one. Due to this impurity of local updates, when we preform aggregation in $block_{t+2}$, we should also take the global model outputted in previous blocks into consideration. As a result, we use the exponential weighted moving average (**EWMA**) of the global model in Fiesta model aggregation. Formally, in $block_{t+1}$, the miner first aggregates the model updates within this block same as FedAvg, and gets the temporary model, say $\mathbf{w}_{t+1}$. Then, the global model will be:

$$\mathbf{w}^{global}_{t+1} = (1-\alpha)\mathbf{w}^{global}_{t} + \alpha \mathbf{w}_{t+1}, \alpha \in (0,1). \qquad (2)$$

Note that the round notion in the synchronous setting does not apply here anymore, we define *async round* as each time a new block is generated with a new global model within. In addition, if miners receive multiple updates from the same

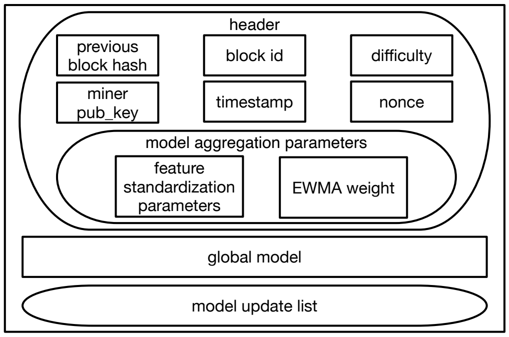


Fig. 3. Block structure of Fiesta.

mobile device in an async round (say $\mathbf{w}_{t-1}^i$ and $\mathbf{w}_t^i$), they should only use the most recent one ($\mathbf{w}_t^i$) for aggregation so that too stale updates will be ignored.

A trivial fact is that in synchronous settings, using EWMA is equivalent to setting a lower learning rate. Yet in asynchronous settings, it is not equivalent considering each block contains model updates from different devices. There are also other ways to penalize staled local updates but EWMA introduces modification with the least complexity.

Block structure. Fig. 3 shows the structure of a block. The principle of block structure design is that all parameters that affect global model aggregation should be recorded in the block, besides standard blockchain required parameters. For example, we include the parameters for feature standardization. Local training related parameters are not included in the blockchain, e.g. optimizer type and the learning rate.

Mining difficulty and block generation speed. In a blockchain, blocks are chained by containing the hash of the previous block and it requires the hash of each block to contain at least d leading zeros, which is the mining difficulty. The mining difficulty also decides the block generation speed accordingly. It balances a tradeoff between the first confirmation time (the time taken to generate the next valid block) and the amount of work wasted due to blockchain forking. A lower mining difficulty means a higher block generation speed, which leads to a higher possibility of blockchain forking and more work wastage but also a shorter time for first confirmation.

Block verification. Before adding the new block received from a neighbor to the blockchain and forwarding it to other neighbors, a miner needs to verify its validity. In addition to checking the PoW validity and block generation time and size, which are similar to bitcoin [23], the followings also need to be verified: a) compatible parameters compared with the previous block, e.g. difficulties and feature standardization parameters. b) the new global model is correctly computed based on the local updates within that block. c) there are no other updates from the same mobile device within a block.

C. Reward mechanisms

Rewards within a certain time period are calculated by the administrator based on the information recorded in the blockchain. Here we only propose a quantification policy. The exact reward could be monetary or using some other credit form. We distinguish two types of rewards, one for individuals and one for organizations.

Individual reward: Previous work on generalized blockchain and federated learning [11] utilizes a reward mechanism that is proportional only to the number of data records each device contributes. However, this approach is not suitable for spectrum sensing. If the reward is purely based on the number of data records, this device will tend to gain more reward without providing much insight into how this channel is utilized across the area, which is undesirable. Thus, we decide that the reward of an update for each mobile device/individual should be proportional to its contribution in loss function minimization. Suppose a model update from client i is trained on local data set D_i with $|D_i| = n_i$, and it uses $block_t$'s global model $\mathbf{w}_t^{global}$ as the initial parameters, and outputs $\mathbf{w}_{t+1}^i$. Denote the average loss on each data record as $l(\mathbf{w}; D_i)$. The individual reward is calculated as follows:

$$r_{t+1}^i = n_i \cdot \left[l(\mathbf{w}_t^{global}; D_i) - l(\mathbf{w}_{t+1}^i; D_i) \right]. \tag{3}$$

Since we assume the mobile clients report these values with the model update to the miners faithfully (this could be enforced by on device trusted execution environment, i.e. TEE), the individual rewards can be calculated reliably at seed node(s). In order to gain more rewards with the same n_i, one can enlarge the difference between $l(\mathbf{w}_t^{global}; D_i)$ and $l(\mathbf{w}_{t+1}^i; D_i)$ by measuring the locations that are not captured by the existing global model rather than visiting the locations that are already measured. This is also beneficial from the entire community's perspective. Furthermore, this reward mechanism also motivates individuals to use appropriate types of optimizer and parameters to decrease the loss quickly, and to have high computational power and low network latency to finish more rounds of local training within a certain time period. Note that the reward is for a fixed number of epochs (as a local round). Increasing reward by conducting local training for too many epochs before uploading can be harmful for convergence [17].

Organization reward: It is desirable that the reward to each organization is proportional to the contribution in the computing power that aggregates updates from mobile devices and generates the new global model. Therefore, the reward to each miner/organization is proportional to the number of blocks generated in the blockchain within the time period, similar to bitcoin.

We envision that the funding for the reward mechanisms can come from two resources, including the administrator itself, and the revenue generated by selling the learned model or the right to query the learned model. The allocation of the funding between the individual reward and the organization reward can be dynamically adjusted to advocate more mobile devices with spectrum sensing capability or more miners. This kind of funding and rewarding ecosystem is actually quite essential for a crowsourced project running in a sustainable way.

D. Joining and leaving Fiesta

Due to the peer-to-peer nature of the blockchain protocol, it is easy to join Fiesta for both individuals and organizations. Given the joining requests from organizations' miners and individuals' mobile devices, the seed node(s) reply with a list of miners they are aware of. The joining miners can then establish connections with a group of other miners. Different from miners, the individuals' mobile devices can select the miner with the smallest latency within this group before it enters the learning mode. As the result, the latency between the mobile device and the miner is minimized. In addition, the mobile devices also can obtain the recommended optimizer type and parameters, the learning rate, and the decay rates of the momentum during their joining process, although not forced to use them. Moreover, the freedom of joining for both organizations and individuals makes sharing the model easy. As long as one is contributing to the system, it inherently has access to the up-to-date global model.

Similarly, miners and mobile devices can leave the system whenever they like, which overcomes the drawback of the single point of failures in ordinary federated learning.

E. Motivation of integrating BlockChain

There are actually abundant solutions to decentralizing a system and enhance its robustness, for example, Paxos [15], Raft [24], and PBFT [6]. But we still choose blockchain [23] (proof-of-work, to be exact) as the consensus algorithm for the following reasons:

Scalability. Conventional consensus algorithm usually has one or several strong leader nodes, either through designation or election. It is not a problem when there are only a few nodes. For example, Google's distributed lock service and file system, Chubby [4], is based on Paxos and consists of only five replicas. However, in the context of crowdsourcing, with more and more nodes joining into the community, communication and coordination cost will inflate dramatically and soon become the bottleneck of those leader nodes' performance [12]. PoW does not suffer from scalability problem that much since all that member node needs to preform is checking the hash value of each block it receives and follows the longest chain in the community [21].

Security. Node failure is not the only threat for robustness. Sometimes malicious nodes deteriorate the system in a more severe way, especially in the context of crowdsourcing where the community is open to everyone. One of the most common attack could be Sybil attack [8] where malicious miner injects overwhelming numbers of forged local updates to gain extra rewards for this dummy contribution. On one hand this could be partially prevented through a stricter local updates verification. For example, a small but generalized test set can be spread by seed node to each miner. A local updates received by a miner node, will be considered as valid if it output limited error, and will be regarded as invalid if the error is beyond a threshold. On the other hand, the nature of PoW ensures that no matter how many malicious nodes one has instantiated (cloud virtual machine, for example, is a cheap way to do so) or how many dummy updates one has injected, it is the computation power that has the impact on final consensus. This mechanism lower the volume of the attacker dramatically and increases their cost of collaring illegal rewards.

Alternative consensus mechanism for blockchain. Another well-known substitution to PoW is Proof-of-Stake (PoS) [10], where it is the earned token instead of the computation power that supports the volume of each node. But PoS suffers from the nothing-at-stake problem severely [16] and has not been widely tested. Hence we simply pick PoW in our implementation for our current implementation.

III. IMPLEMENTATION & DEPLOYMENT

We implement the three roles of Fiesta's layered architecture using different hardware and software stack.

Mobile device. A mobile device is connected with an external sensor board through a USB cable to gain spectrum sensing ability, which is shown in Fig. 4(a). The sensor board we use is from [18]. It is equipped with an AD9361 RF transceiver and a ZYNQ-7020 FPGA chip, featuring strong spectrum sensing capability ranging from 70MHz to 6GHz with a 56MHz sampling rate and high programmability. Users can check the status and interact through an Android app, as shown in Fig. 4(a). The app will enter the **sensing mode** when a sensor board is attached, and will start the **machine learning mode** when the mobile device is in charging and under stable WiFi access. This dual mode design (or lazy training design) dramatically reduces the impact of our spectrum sensing task onto the daily usage of host device. In terms of the software stack, usb-serial-for-android and deeplearning4j are integrated into the Android app to facilitate serial communication with the sensor board and on-device machine learning.

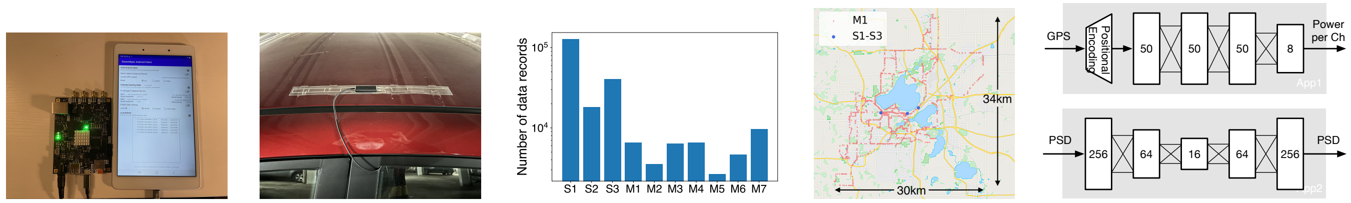
Miner. Miners are realized using Flask, a lightweight web application framework based on Python, and all the communications between mobile devices and miners use HTTP protocol for simplicity. An ordinary blockchain node using Proof-of-Work (PoW) consensus algorithm written in Python consists of the main body of each miner. Having received several local weight updates from mobile devices, the miner will integrate them into a new block and compute the new global model, for the next async round of training. For our experiments and deployment, several miner servers are deployed on different host machines including a 16-core 16 GB RAM personal desktop and an 8-core 8 GB RAM lab workstation.

Seed node. Seed node is also a Flask server and it uses HTTP to communicate with other roles as well. The seed node is backed up with two tables, one noting down the members registered for Fiesta and the other recording the contribution and reward for each member. In our design, there are a limited number of seed nodes and each of them should be reserved for only the administrator. For our experiments and deployment, a seed node is running on a 16-core 16 GB RAM desktop with a public domain name to allow other roles to query the seed node and register themselves through the Internet.

A. Implementation Challenges

We resolve several issues during implementation to make Fiesta's implementation efficient and practical to be deployed.

Local update suppression. When the model converges and the pattern of the local data is already learned, the mobile device should not upload the local model to save the network bandwidth of both the mobile device and the miners as well as the computing resources at the miners. Given the reward



(a) Android app with the sensor board.

(b) Antenna mounted on a vehicle.

(c) No. of records collected by each device.

(d) Distribution of measurement location.

(e) Learning models of our tasks: MLP with freq positional encoder for App1 and an autoencoder for App2

Fig. 4. HW/SW suite, real world deployment, and demo tasks of Fiesta.

mechanism we propose, this can be implemented as the mobile device stops uploading the update if the individual reward calculated based on local training loss reduction is smaller than a minimal threshold, i.e. when the local training loss does not decrease significantly, or the reward is non-positive.

Oversized models. Model size can be large, which makes block size large. This leads to two drawbacks: a heavy burden on the network transmission during miner communication, and a burden on the computation workload to the hash function when performing PoW. To address the former, we only keep full information of a block in its born miner and another archive server (which could be seed node), and share a streamlined block keeping only necessary components. To address the latter, we perform a first stage hash over those local updates from each different mobile devices, and use the generated fixed-length hash string to further generate the hash of the whole block when finding the nonce during PoW. As a result, hashing of a long model is replaced with hashing a shorter and fixed length string.

Model structure evolution. The structure of the model may need to grow more complex to better fit the increasing number of data records. At this time, seed node(s) will send out a broadcast message with their signature, which then will be verified by miners and other seed node(s), such that the miners start to work on a new blockchain upon receipt of this message. The genesis block of the new blockchain contains the new model structure. In order to fully utilize the data already obtained, there could be a knowledge transfer process before announcing the new structure, which means the global model in the genesis block of the new blockchain may not be randomly initialized.

B. Deployment and Applications

We deploy 10 sensor boards with mobile devices in a mid-sized US city for 3 months. 3 sensor boards are static (S1-S3), and the rest are deployed on personal vehicles of volunteers (M1-M7). Fig. 4(b) shows an antenna mounted on a vehicle. Due to the frequency constraint of the antennas, we configure the sensor boards to capture Power Spectrum Density (PSD) readings of 256 bins from 500MHz to 800MHz on a 50MHz basis, among which 500-700MHz is primarily for TV broadcast and 700-800MHz is mainly for LTE usage. PSD data is collected at the rate of one data record per minute for each 50MHz band. Fig. 4(c) shows the number of data records per sensor. Overall, we have gathered more than 220K data records of 600 hours with significant heterogeneity among different mobile devices. Fig. 4(d) illustrates the locations of measurements. It shows the static nodes and the GPS trace of one mobile node, only for illustration.

We have deployed two different machine learning applications in real world evaluation to demonstrate the functionality of our design. Network structure of the machine learning models are illustrated in Fig. 4(e). Both application leverage multilayer perceptron (MLP) as basic building block, where digits on each block refers to how many neurons in that layer.

SpatialSensing(App1): This application is to learn how the TV spectrum is used in the spatial domain through a neural network model predicting the signal power of TV channels at different locations. The model is a feed-forward neural network with latitude and longitude as inputs and signal power of TV channels as output, with mean square error as the loss function. In the actual implementation, a sin-cos positional encoding is adopted, inspired by NeRF [20]. Specifically, the GPS location, after standardization, will be fed to an encoding function and projected to higher dimension as a longer feature vector, through which the model's fitting capacity will be enhanced dramatically. The encoding function could be mathematically represented as followed:

$$\begin{aligned}\gamma(x,y) = [&sin(\pi x), cos(\pi x), sin(\pi y), cos(\pi y), \ldots,\\ &sin(2^{L-1}\pi x), cos(2^{L-1}\pi x), sin(2^{L-1}\pi y), cos(2^{L-1}\pi y)]\end{aligned} \tag{4}$$

x, y represents normalized GPS latitude and longitude. γ represents the GPS embedding that is about to be sent to the neural network. L here is the hyperparameter that decides the depth of embedding and we empirically choose 10. In the future, environment information including weather condition could be fed into the model as an additional input to enhance model accuracy.

AnomalyDetection(App2): This application is learning how the LTE band is used in the frequency domain by training an autoencoder [3] on the PSD data. The autoencoder is basically a neural network with symmetric structure design and hourglass shape. It takes the PSD data as input and the output of the autoencoder tries to reconstruct the input with high fidelity. The reconstruction result is expected to function as a precursor for detecting frequency domain anomalies, similar to [26].

IV. EVALUATION AND RESULTS

We first benchmark Fiesta's performance by simulations using available spectrum datasets from [30], which contains

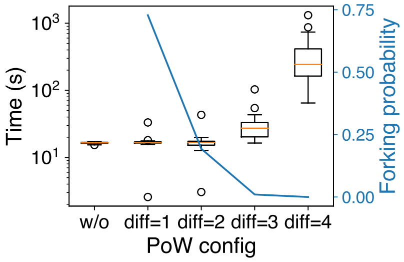
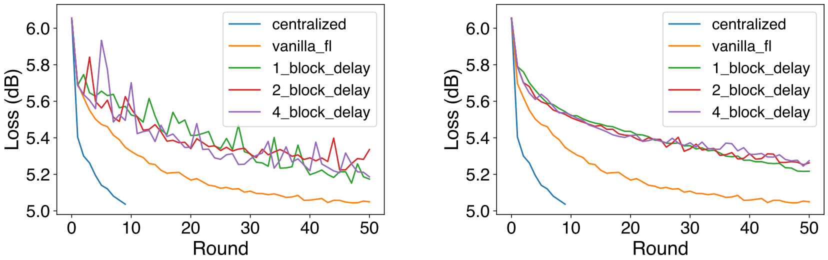


Fig. 5. Impact of mining difficulty: higher difficulty comes with higher PoW overhead, yet in this case the blockchain forks less frequently.



(a) Without EWMA **(b)** With EWMA

Fig. 6. Impact of model aggregation with EWMA on model convergence and accuracy. The model with EWMA converges with much less oscillation.

TV band PSD data with spatial variations in a city. We also demonstrate Fiesta's robustness through node failure and Sybil attack simulation on the miner side, after which we report the evaluation results of our real-world deployment in a mid-size US city gathered by 10 mobile devices for 3 months.

A. Microbenchmarks

In order to verify the correctness and robustness of Fiesta, we use the following simulation setting. We have 5 miners and 10 mobile devices in the system, which is the same as the real-world deployment. The learning mode of mobile devices is simulated through Python code. One round of local training in a mobile device contains 10 epochs. All miners and mobile devices processes run in parallel in a single 28-core machine so that each process occupies roughly the same hardware resource, i.e. one core. We use **SpatialSensing** for benchmarking. For simplicity, we only use the readings of channel 19 (500-506 MHz) in [30]. The data are randomly sampled with 10% probability as the validation set first. For the rest 90%, they are sorted by timestamp and then allocated to the mobile devices as 10 consecutive time periods of a same number of data records. Although the number of data records on each mobile device is the same, there is a certain degree of heterogeneity in terms of spatial distribution.

In the following subsections, "baseline" refers to the naive synchronous FedAvg. consisting one aggregation server and 10 mobile devices, labeled as "w/o" in Fig. 5 and as "vanilla_fl" in Fig. 6. The label "centralized" refers to local training with the full dataset in a contrast to the federated learning fashion.

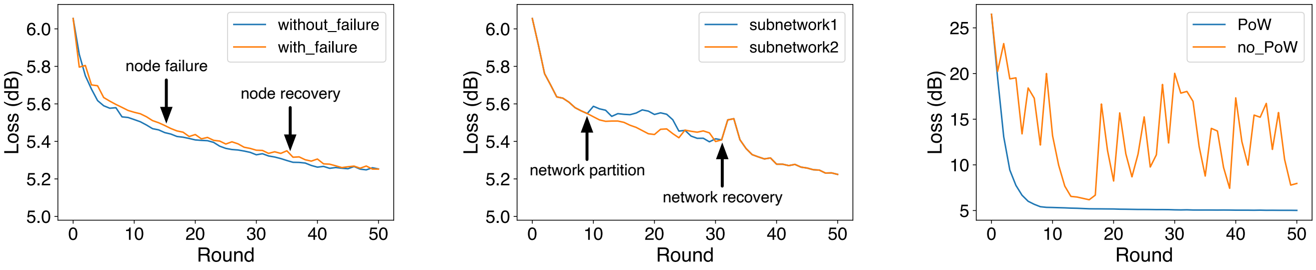
1) Mining difficulty: We first decide the mining difficulty to balance the first confirmation time and the forking probability of blockchain. The difficulty d means, in order to generate a valid block, the number of leading zeros of the block hash must equal d. We calculate forking probability as the total number of block replacements occurring due to knowing a longer chain from other miners divided by the total number of blocks in the blockchain. Here, we use the synchronous setting , i.e. a miner must collect local updates from all of the 10 mobile devices before generate the new block, to simplify the configurations.

We vary the difficulty and compare it with the baseline, where PoW is disabled in the system. We measure the first confirmation time, which is defined as the time period between the generation of two consecutive blocks in the blockchain, and the forking probability. Fig. 5 shows the results. When the PoW is disabled, we can see that the first confirmation time is 11 seconds, without many variations in the distribution. The major factor of this first confirmation time is attributed to the time spent in the mobile devices for a round of local training. Network latency is negligible in this case. When mining difficulty < 2, although there are some outliers, the distribution of the first confirmation time does not change much compared with the scenario when PoW is disabled. However, when the difficulty > 2, the first confirmation time starts increasing dramatically. In this setup, time spent on mining for a valid block becomes substantial. Moreover, the distribution of the latency spreads more widely, compared with the baseline. As for the result of forking probability, we can see from Fig. 5, if the difficulty is no more than 2, we have a significant non-zero possibility of forking. Nevertheless, if the difficulty is greater than 2, we can reduce the forking probability to a value close to zero.

Considering the results of first confirmation time and forking probability, we fix the difficulty as 3 for the rest of our simulations and the real-world deployment. Note that forking probability is also related to the scale of the miner network. The larger scale of the miner network, the higher the forking probability it can get. We use a fixed difficulty rather than a dynamic one because the scale of our miner network does not increase significantly during the real-world experiment.

2) Effectiveness of EWMA: We study the effect of model aggregation with EWMA introduced in § II-B on model convergence and accuracy. In detail, we let the blockchain contain 5 model updates in each block precisely and divide the 10 mobile devices into two groups. There are 5 faster mobile devices that upload their updates immediately after their local training round is finished, i.e. no block delay. The other 5 mobile devices are the slower ones, whose updates are at least $b (b >= 1)$ blocks delayed (specified in the legend of Fig. 6. We use the loss on the validation set to check the convergence and accuracy.

Fig. 6 illustrates the results. If EWMA is not used when doing model aggregation, we can see from Fig. III-B that the loss would fluctuate significantly. Fig. III-B shows that if EWMA is used when doing model aggregation, the learning curves are very smooth without sacrificing accuracy, compared with not using EWMA. Having large fluctuations in the learning curve is even more undesirable if we consider the possibility of mobile devices joining/leaving, the heterogeneous data distribution on each device, and new data collected. Additionally, Fig. 6 also contains the results of centralized learning (marked as "centralized") and vanilla federated learn-



(a) Miner node failure. (b) Miner network partition failure. (c) Loss when exposed to malicious attack.

Fig. 7. Impact of failures and malicious attacks on model training.

ing (marked as ”vanilla_fl”) in the synchronous setting. Vanilla federated learning converges slightly faster than the results of the asynchronous settings and has a marginally better estimation accuracy.

Note that α in EWMA is set as 0.5 for all the simulations and the real-world deployment to balance past and present, which can be further fine tuned if needed.

3) Robustness: We study the robustness introduced by blockchain, which does not exist in centralized learning and ordinary federated learning. We use an asynchronous setting where each block contains 4˜10 model updates with EWMA enabled.

We first study system’s resistance toward miner node failure. Fig. 7(a) illustrates the validation loss as a function of the number of rounds with miner node failure and recovery. A randomly chosen miner node fails after round 15 is finished and is back online after round 35 is finished. In Fig. 7(a), the curve standing for the node failure experiment converges to a similar accuracy at approximately same speed compared to the non-failure one.

By using a distributed miner network, another type of network failure is possible, which is network partitions. The miner network is partitioned into two subnetworks after the birth of block 10, with one subnetwork containing 2 miners and the other containing 3 miners. After block 31, the two subnetworks merge back into one. Further, the validation loss still gradually goes down, almost the same as the no failure case in Fig. 7(a). We admit that limited failure rehearsals don’t speak loudly enough for a perfect robustness, but it does demonstrate Fiesta’s enhancement in its resistance towards failures in a qualitative manner.

As mentioned in § II-E, we select PoW among other kinds of consensus protocols out of scalability and security consideration. Fig. 7(c) shows an attack simulation based on synchronous setup, where one (call it malicious node) of those miners keeps injecting blocks that contain random generated local updates to its own blockchain then spread it. This attack simulation uses a randomly initiated model (instead of a lightly pretrained model like other experiments) as the initial global model, hence the initial loss is relatively high (over 25 dB). Compared with a plain fully-replicated distributed system who fails to converge (labeled as ”no_PoW”), Fiesta (labeled as ”PoW”) could still achieve model convergence with PoW. Note that we do apply the aforementioned stricter local update check in both setup and good miners will reject bad local updates spread from that malicious miner, but they still can not prevent malicious miner inject overwhelming numbers of deteriorated block into the main chain with the absence of PoW.

B. Real-world evaluation results

We deploy Fiesta onto Android devices with sensor boards, along with two seed node and three miner nodes for each App. These devices present a wide variety in terms of their models (tablets and phones) and mobility status (stationary or portable), which cover the majority of real-world scenarios.

1) Overhead on mobile devices: Machine learning on mobile devices may be of low efficiency and time-consuming. Hence, profiling of its overhead is necessary.

CPU and memory. Fiesta Android client occupies less than 200MB RAM, and 50% CPU utilization on Google Pixel 2 when training two models. On the other hand, in sensing mode instead of machine learning mode, the CPU utilization is negligible considering the only workload is receiving data from the sensing board. Considering Pixel 2 is a device released 7 years ago, powerful tensor accelerators like Google G2 in today’s smart phones have the potential to bear the training task and further reduce the CPU workload.

Network traffic. In order to fetch the global model from a miner and upload the trained local model to a miner, data packets sized 220KB for **SpatialSensing** and 1.5MB for **AnomalyDetection** will be exchanged during one local round of training.

Training speed. A local training round (contains 10 epochs) with a training set with 4k data records takes 1.3 minutes for **SpatialSensing** and 2 minutes for **AnomalyDetection** on Samsung Galaxy Tablet A 8.0, and slightly faster on Google Pixel 2. The difference between applications comes from the variance of their neural network structures. We also observe that one round of training time is proportional to the size of the local dataset and the number of epochs.

Energy consumption. Since local training is triggered only when the device gets plugged with a charger, energy consumption in machine learning mode is no big issue. Yet we still test energy consumption based on battery. Google Pixel 2 whose battery has a capacity of 2700mAh can hold up to 5 hours of continuous training, while Samsung Galaxy Tablet A 8.0, with a 6250mAh battery could support up to 8 hours. For sensing mode, where the device is linked with a sensor board, tablets can last for 4 hours and only 2 hours for Google Pixel 2, because not only the Android device needs to power

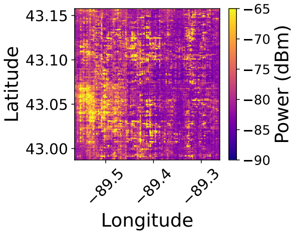


Fig. 8. SpatialSensing: Fiesta distributedly learnt TV Ch. 46 power heatmap.

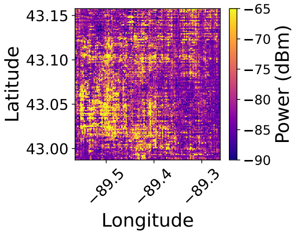


Fig. 9. SpatialSensing: centralized learnt TV Ch. 46 power heatmap.

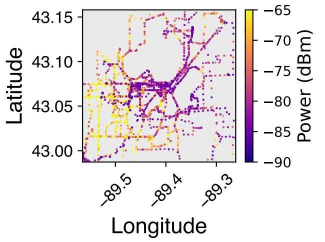


Fig. 10. SpatialSensing: ground truth of TV Ch. 46 power heatmap.

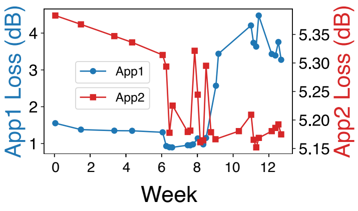


Fig. 11. Model convergence across the time: App1-SpatialSensing, App2-AnomalyDetection

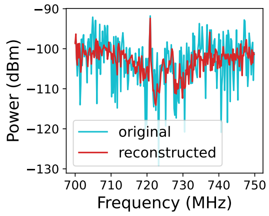


Fig. 12. AnomalyDetection: example of reconstructed PSD data.

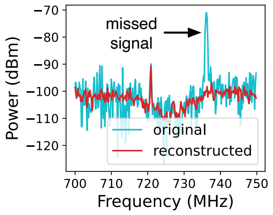


Fig. 13. AnomalyDetection: example outlier of LTE band.

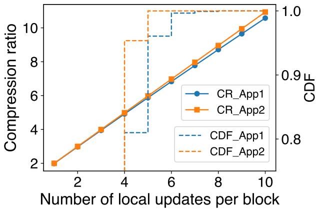


Fig. 14. Compression and distribution of block size.

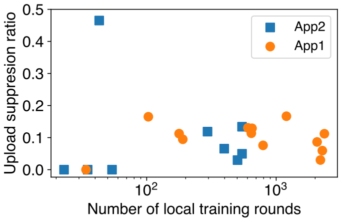


Fig. 15. Upload suppresion ratio.

the board through OTG, GPS service also drains significant energy. The remarkable power consumption of both training and data sampling is one of the intuition why we separate sensing mode and learning mode in § III.

We believe the power drain in sensing mode could be further reduced: first, with the development of Minimization of Drive Tests (MDT), we could expect modern devices to be integrated with a more versatile radio frontend with wider spectrum sensing capacity and precision. This could alleviate the necessity of a dedicated and energy hungry FPGA based dongle. We could also offload signal processing onto the on-device computation capacity like DSP and NPU. Second, we could also reduce the energy cost of GPS by using a combination of sparse GPS localization and IMU [2].

2) Convergence and learned models of the two applications: We track the convergence during the whole federated learning procedure. Unlike an off-the-shelf training set with a fixed validation set in our microbenchmark simulation, for the real-world deployment, it is hard to evaluate the trained model directly, considering the training set itself is gathered in a streaming manner. On the other hand, a neat and distinct definition of ”one round of federated learning” is impossible due to this asynchronous setup. As a compromise, we define *AppLoss* using the weighted average of training loss (MSE) among local updates within $block_t$.

$$AppLoss = \frac{1}{\sum_{i \in block_t} n_i} \sum_{i \in block_t} n_i \cdot l(w_t^i; D_i). \qquad (5)$$

$l(w_t^i; D_i)$ is the loss after the last epoch of the local training round from mobile device i with local dataset D_i, where $|D_i| = n_i$.

After calculation of the *AppLoss* for each block, “convergence curves” as functions of date for both apps are shown in Fig. 11. The convergence curves only visualize a subset of data points for clarity.

Before week 6, there are only 5 sensor nodes deployed, including 3 static and 2 mobile. We require each block contains 4∼10 local updates. Reduction of both apps’ losses is observed.

Starting from week 6, five additional sensor nodes are deployed and the losses got further reduced. With more devices joining, the asynchronous nature of this bigger system emerges and leads to inconsistent compositions between blocks, reflected as noticeable fluctuations afterward.

A distinct leap occurs for **SpatialSensing** loss after week 8 because we online upgraded the machine learning model through transfer learning and the output dimension was increased from single one to eight causing a higher MSE loss. Detail is discussed in § IV-B3. Finally, the losses of **SpatialSensing** and **AnomalyDetection** converge to 4dB and 5dB respectively.

We are also interested in what those models have learned. Fig. 8 visualizes the estimated power of TV channel 46 (662-668 MHz). The figure is generated through sampling **SpatialSensing** model with a resolution of 100×100 within a specific latitude and longitude range. To intuitively demonstrate that the model has learnt the geological feature of spectrum usage successfully, Fig. 10 visualize the raw data (available in kaggle) we collected from user devices directly. From either heat map and combining our prior knowledge with the city’s building distribution, we notice that area with a higher density of high-rise buildings tends to have lower TV power and vice versa.

We also reconstruct the heat map through traditional centralized learning instead of federated learning way in Fig. 9, based on all of the gathered data. A general similarity among Fig. 8, Fig. 9, and Fig. 10 demonstrates the feasibility of using distributed federated learning in replacement of centralized learning or even raw data, while achieving good fidelity. A mismatch of regional details is also acceptable considering the raw data itself only covers main streets in the city (Fig. 4(d))

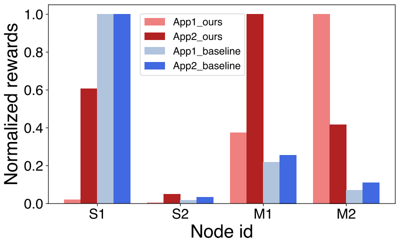


Fig. 16. Comparison of different reward policies.

instead of every square feet. As a reference, test loss for **SpatialSensing** in the centralized setup converges to 2.5 dB.

Fig. 12 shows the reconstruction result of a sample without obvious signals using the autoencoder trained for **AnomalyDetection**. The non-flat noise floor of the sensor including a spike around 720MHz is successfully revealed after reconstruction. Fig. 13 shows a signal not fully reconstructed by the model, which might be a frequency domain outlier and could be used for further anomaly detection. Other spike due to non-flat noise floor of the sensor is still successfully reconstructed.

3) Model structure evolution: On week 9, we online upgrade the structure of the model for **SpatialSensing** to output multiple TV channels' power instead of only one. We achieve knowledge transfer by duplicating the original single output neuron to initiate other output neurons corresponding to the multiple TV channels, and keep the hidden layers' parameter unchanged in the meantime. The MSE loss increases due to a larger output dimension. Furthermore, estimating multiple TV channels' power is intrinsically more challenging compared with a single one. Fig. 11 shows the ease of use of Fiesta when one needs to modify the machine learning model structure.

4) Effectiveness of model update suppression: As mentioned in § III, when the pattern of local data has already been fully learned, i.e. when the local loss does not decrease significantly, mobile devices will retreat from uploading. Fig. 15 shows how frequently this kind of upload suppression happens among devices. From Fig. 15 we observe that for the mobile devices who have trained their gathered data more times, the upload suppression scheme is more likely to be triggered. This is desirable since information within those data has already been fully learnt. Note that the opposite may not hold considering when a mobile device that rarely trains the data it gathered, the knowledge within may already be obtained from another device with a large amount of data and frequently performs training. The outlier point on the left top of **AnomalyDetection** in Fig. 15 is an example of this case.

5) Effectiveness of the mechanism to deal with oversized models: Machine learning models with thousands or even millions of parameters will be a huge burden for the network and local storage. Fig. 14 shows our block compression policy with respect to the number of local updates within a block. Considering we replace the giant Python dictionary of each local weight with a much smaller unit (§ III-A), the more local updates a block contains, the more storage we could save.

We also show the CDF of the number of local updates within a block. We notice that due to the relatively small mining difficulty, most blocks contain only 4 local updates, the minimal number we set. The max number of local updates a block includes is 8. **SpatialSensing** blocks contain more local updates than **AnomalyDetection** blocks. It is reasonable considering the machine learning model of **AnomalyDetection** contains much more parameters than that of **SpatialSensing**, hence requiring a longer time for local training. More swiftly the local training completes, more local updates can be included in one block given the mining difficulty.

6) Reward mechanisms: As mentioned in § II-C, the reward mechanism previous work [11] adopts fails to reflect the actual contribution of each member. To demonstrate the effectiveness of our proposed reward mechanism shown in formula 3, we compare it with the naive size-based reward mechanism as the baseline, in Fig. 16, using 4 representative devices, including two stationary and two mobile. All the reward values are normalized by the maximum.

Stationary device 1 performs local training on a large but roughly geographically homogeneous dataset considering it is deployed in a stationary environment. On the contrary, mobile device 1 and 2 have relatively smaller but more heterogeneous data because they are deployed on moving vehicles. From Fig. 16, we observe that the baseline mechanism grants higher rewards of the stationary device regardless to the little information within, while the proposed mechanism values the data diversity significantly. The difference is more distinct when it comes to **SpatialSensing**, which tries to learn the spatial distribution of TV channels' power. Stationary device S2, though having more data points than mobile devices, but performs few local training and uploading, hence little reward earned. One may also notice that stationary device S1 obtains much more reward than S2, although it collects less data as shown in Fig. 4(c). This is because S2 rarely trains the model and has submitted merely dozens of local updates along with the number of local data records, while S1 contributes over thousands of updates.

Compared with the baseline reward policy who quantifies the contribution with the absolute number of collected data records, our proposed reward policy encourages more informative contributions.

V. DISCUSSION AND FUTURE WORK

Privacy leakage. Although the data does not leave the mobile devices, there is still a possibility of privacy leakage in federated learning. Previous global models and the gradient updates from an individual can be used to infer the local training data held by that user [9], and these two pieces of information are available in Fiesta. Recent work on privacy-preserving federated learning [22] advocates running local training at mobile devices and secure model aggregation at servers in Trusted Execution Environments to reduce leakage. Nevertheless, whether it is fully compatible with Fiesta, especially with the openness and the joining freedom of the miners, still needs further exploration.

Adversarial robustness. In this work, we have assumed that the mobile clients function with no malicious intent. Although it is a valid assumption currently given that we have full control of the hardware and software being used, it is probably no longer valid as the deployment scale enlarges. Taking advantage of the openness of Fiesta, malicious individuals can introduce data poisoning attacks and model updates can also be compromised by malicious miners. The aforementioned local update check might mitigate the issue but achieves very limited distinguishing capability with an extra cost of spreading and storing test set, and how to deal with malicious participants in general remains an open question in the federated learning research area [9].

Model inference. Although in this paper we mainly focus on model learning, model inference is naturally supported by Fiesta, given that model inference is just a forward propagation during the training process. Not only model inference is much less computationally intensive, but it is also more well-supported than model training in popular machine learning frameworks e.g. PyTorch and TensorFlow on current mobile devices. We envision these popular frameworks will continue to evolve and better support model training on mobile devices in the future. Moreover, before model inference is enabled at mobile devices, the learned global model may need to be personalized due to non-IID distribution of data [14] for better performance.

Alternative to the external SDR dongle. The current implementation of Fiesta requires an external FPGA-based SDR dongle to perform spectrum sensing tasks. However, this approach introduces concerns regarding additional energy consumption and practical inconvenience for daily usage. To address these limitations, we envision future implementations that extract CSI information for spectrum sensing directly within smartphone chipsets. This approach has been demonstrated in WiFi chipsets through the NextMon project [28], and a similar methodology could be adapted for 5G signals. One promising approach would leverage signals broadcast by default from gNBs, such as synchronization signals (SSB/PSS/SSS). By aggregating information from both the ISM WiFi band and licensed cellular bands, this approach could enable practical spectrum sensing capabilities across all mobile devices without requiring external hardware.

Application to other domains. We demonstrate Fiesta's efficacy on two PSD-based applications formulated as self-supervised regression problems using gradient descent optimizers. IQ data applications are also supported under the same formulation. Classification problems (e.g., signal type classification) could be supported via semi-supervised learning, requiring some labeled data from administrators or organizations with sufficient data collection capacity and domain expertise. Clustering problems are naturally supported without labeling requirements, though interpreting results requires domain expertise and validation on raw data samples. While some techniques are optimized for mobile spectrum sensing, they can be applied to other crowdsourcing scenarios, including static spectrum sensing or non-spectrum domains.

VI. CONCLUSION

A major pitfall of previous spectrum crowdsensing efforts is the lack of awareness of sustainability. We have presented Fiesta, a sustainable framework for spectrum crowdsensing based on a three-tier architecture. On the technical side, Fiesta addresses two deficiencies of prior spectrum sensing architectures: privacy concerns that require participants to upload sensitive information and a single point of failure due to centralized analysis, through a combination of federated learning and blockchain. On the policy side, we have proposed a fair reward quantification mechanism for participants and discussed potential funding sources to ensure long-term sustainability. We also release the code and dataset used in this paper to welcome further contributions and wider adoption from the community to realize Fiesta's long-term vision.

ACKNOWLEDGEMENT

We thank Hongyi Wang for discussions about federated learning in the early stage of the project. All UW-Madison authors were supported in part by the US National Science Foundation awards 2003129, 2112562, 2107060, 2212688, and through an award from the US Department of Commerce with award number 70NANB21H043. The work at IMDEA has been funded by project PID2022-136769NB-I00 funded by MCIN/AEI /10.13039/501100011033 / FEDER, UE.

REFERENCES

[1] Microsoft spectrum observatory. spectrum-observatory.cloudapp.net.

[2] S. Y. Alaba. Gps-imu sensor fusion for reliable autonomous vehicle position estimation. *arXiv preprint arXiv:2405.08119*, 2024.

[3] J. An and S. Cho. Variational autoencoder based anomaly detection using reconstruction probability. *Special Lecture on IE*, 2(1):1–18, 2015.

[4] M. Burrows. The chubby lock service for loosely-coupled distributed systems. In *Usenix OSDI*, pages 335–350, 2006.

[5] R. Calvo-Palomino, H. Cordobés, M. Engel, M. Fuchs, P. Jain, M. Liechti, S. Rajendran, M. Schäfer, B. Van den Bergh, S. Pollin, et al. Electrosense+: Crowdsourcing radio spectrum decoding using iot receivers. *Computer Networks*, 174:107231, 2020.

[6] M. Castro, B. Liskov, et al. Practical byzantine fault tolerance. In *Usenix OSDI*, volume 99, pages 173–186, 1999.

[7] E. Chai, K. Sundaresan, M. A. Khojastepour, and S. Rangarajan. Lte in unlicensed spectrum: Are we there yet? In *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking*, pages 135–148, 2016.

[8] J. R. Douceur. The sybil attack. In *International workshop on peer-to-peer systems*, pages 251–260. Springer, 2002.

[9] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

[10] A. Kiayias, A. Russell, B. David, and R. Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Annual international cryptology conference*, pages 357–388. Springer, 2017.

[11] H. Kim, J. Park, M. Bennis, and S.-L. Kim. Blockchained on-device federated learning. *IEEE Communications Letters*, 24(6):1279–1283, 2019.

[12] M. Kogias and E. Bugnion. Hovercraft: Achieving scalability and fault-tolerance for microsecond-scale datacenter services. In *Proceedings of the Fifteenth European Conference on Computer Systems*, pages 1–17, 2020.

[13] J. Konečnỳ, B. McMahan, and D. Ramage. Federated optimization: Distributed optimization beyond the datacenter. In *NIPS Optimization for Machine Learning Workshop*, 2015.

[14] V. Kulkarni, M. Kulkarni, and A. Pant. Survey of personalization techniques for federated learning. In *2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4)*, pages 794–797. IEEE, 2020.

[15] L. Lamport. The part-time parliament. In *Concurrency: the Works of Leslie Lamport*, pages 277–317. 2019.

[16] C. Lepore, M. Ceria, et al. A survey on blockchain consensus with a performance comparison of pow, pos and pure pos. *Mathematics*, 8(10):1782, 2020.

[17] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang. On the convergence of fedavg on non-iid data. In *ICLR*, 2020.

[18] Y. Li, Y. Zeng, and S. Banerjee. Enabling wideband, mobile spectrum sensing through onboard heterogeneous computing. In *Proceedings of the 22nd International Workshop on Mobile Computing Systems and Applications*, pages 85–91, 2021.

[19] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282. PMLR, 2017.

[20] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

[21] D. Mingxiao, M. Xiaofeng, Z. Zhe, W. Xiangwei, and C. Qijun. A review on consensus algorithm of blockchain. In *2017 IEEE international conference on systems, man, and cybernetics (SMC)*, pages 2567–2572. IEEE, 2017.

[22] F. Mo, H. Haddadi, K. Katevas, E. Marin, D. Perino, and N. Kourtellis. Ppfl: Privacy-preserving federated learning with trusted execution environments. In *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '21, page 94–108, New York, NY, USA, 2021. Association for Computing Machinery.

[23] S. Nakamoto. Bitcoin whitepaper. *URL: https://bitcoin. org/bitcoin. pdf*, 2008.

[24] D. Ongaro and J. Ousterhout. In search of an understandable consensus algorithm. In *Usenix ATC*, pages 305–319, 2014.

[25] S. Rajendran, R. Calvo-Palomino, M. Fuchs, B. Van den Bergh, H. Cordobés, D. Giustiniano, S. Pollin, and V. Lenders. Electrosense: Open and big spectrum data. *IEEE Communications Magazine*, 56(1):210–217, 2017.

[26] S. Rajendran, W. Meert, V. Lenders, and S. Pollin. Saife: Unsupervised wireless spectrum anomaly detection with interpretable features. In *2018 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*, pages 1–9, 2018.

[27] S. Sarkar, M. Buddhikot, A. Baset, and S. K. Kasera. Deepradar: a deep-learning-based environmental sensing capability sensor design for cbrs. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, pages 56–68, 2021.

[28] M. Schulz, D. Wegemer, and M. Hollick. Nexmon: The c-based firmware patching framework, 2017.

[29] Y. Zeng, R. Calvo-Palomino, D. Giustiniano, G. Bovet, S. Banerjee, et al. Adaptive uplink data compression in spectrum crowdsensing systems. In *IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*, 2021.

[30] Y. Zeng, V. Chandrasekaran, S. Banerjee, and D. Giustiniano. A framework for analyzing spectrum characteristics in large spatio-temporal scales. In *The 25th Annual International Conference on Mobile Computing and Networking*, pages 1–16, 2019.

[31] Y. Zeng, B. Liu, Y. Li, D. Giustiniano, S. Banerjee, et al. Sustainable spectrum crowdsensing. In *IEEE International Symposium on Dynamic Spectrum Access Networks*, 2024.

[32] T. Zhang, N. Leng, and S. Banerjee. A vehicle-based measurement framework for enhancing whitespace spectrum databases. In *Proceedings of the 20th annual international conference on Mobile computing and networking*, pages 17–28. ACM, 2014.

[33] M. Z. Zheleva, R. Chandra, A. Chowdhery, P. Garnett, A. Gupta, A. Kapoor, and M. Valerio. Enabling a nationwide radio frequency inventory using the spectrum observatory. *IEEE Transactions on Mobile Computing*, 17(2):362–375, 2017.

Yijing Zeng Yijing Zeng is a senior research scientist at Meta, currently working on network infrastructure for AI. He received his bachelor degree in Communication Engineering from Beijing University of Posts and Telecommunications in 2012, his master degree in Electrical and Computer Engineering from University of California-Santa Barbara in 2015, and his PhD in Computer Sciences from University of Wisconsin-Madison in 2022. He contributed to this paper while he was a Phd student at University of Wisconsin-Madison.

Bangya Liu Bangya Liu is a PhD student at the University of Wisconsin-Madison. His research interests lie in mobile computing and spectrum sensing.

Yilong Li Yilong Li is a PhD student at the University of Wisconsin-Madison. His research interests include wireless systems and software-hardware co-designed systems.

Domenico Giustiniano Domenico Giustiniano is a Research Professor (tenured) at IMDEA Networks, Madrid, Spain. He holds a PhD in Telecommunication Engineering from the University of Rome Tor Vergata (2008). Before joining IMDEA, he was a Senior Researcher and Lecturer at ETH Zurich. He also worked for four years as Post-Doctoral Researcher in Disney Research Zurich and Telefonica Research Barcelona. He has authored over 150 international papers, he is Leader of the OpenVLC Project and Co-Founder of the non-profit Electrosense Association. His current research interests cover Battery-Free IoT, Large-scale Spectrum Analytics, Unmanned Aerial Networks, and 6G Localization Systems.

Suman Banerjee Suman Banerjee is the David J. DeWitt Professor of Computer Sciences at the University of Wisconsin-Madison and the founding director of the WiNGS laboratory, which broadly focuses on research in wireless and mobile networking systems. He has published more than 100 technical papers in leading conferences and journals. He received his undergraduate degree from IIT Kanpur, and MS and PhD degrees from the University of Maryland. He is the inaugural recipient of the ACM SIGMOBILE Rockstar award and a recipient of the US National Science Foundation Career Award. He served as the chair of ACM SIGMOBILE from 2013 to 2017. He is a fellow of the ACM and of the IEEE.