Optimization of User Resources in Federated Learning for Urban Sensing Applications

Sparsh Gupta, Ayshika Kapoor & Dheeraj Kumar*
sgupta7@ee.iitr.ac.in,ayshika_k@ece.iitr.ac.in,dheeraj.kumar@ece.iitr.ac.in
Department of Electronics & Communications Engineering, Indian Institute of Technology Roorkee
Roorkee, Uttarakhand, India

ABSTRACT

Participants involved in federated learning based urban sensing tasks are prone to unknowingly leak sensitive information to the central server or an adversary. Secure multiparty computation is a promising solution for protection against inference attacks without compromising on application model accuracy. However, existing secure multiparty computation protocols are resource intensive as they require multiple communication and computation rounds between participants and the central server. For urban sensing applications, where application model is usually a spatiotemporal map over a large area, this is an even more challenging problem to deal with. To achieve real time sensing using frequent model updates, existing state-of-the-art secure multiparty computation protocols such as Turbo-Aggregate shall not be feasible. This paper presents an optimised Turbo-Aggregate protocol, we call Resource Adaptive (ReAd) Turbo-Aggregate, which is a secure multiparty computation scheme designed specifically for urban sensing applications where different participants have varying computation and communication resources. It is an adaptive scheme which grants the flexibility of modifying the space and time granularity of the application model in each round of aggregation to suit the participants' network, processing, and battery resources while retaining the features and security of the parent Turbo-Aggregate protocol. The proposed approach is verified with simulation experiments for the noise mapping task of urban sensing applications. The results demonstrate that the proposed solution provides a useful application model along with the benefits of user privacy using limited computation and communication resources of participating users.

KEYWORDS

Federated learning, urban sensing, noise map, Secure multiparty computation, Turbo-Aggregate, resource-adaptive

ACM Reference Format:
Sparsh Gupta, Ayshika Kapoor & Dheeraj Kumar*. 2023. Optimization of User Resources in Federated Learning for Urban Sensing Applications. In *KDD FL4Data-Mining '23, August 7, 2023, Long Beach, CA, USA.*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Over the years, ubiquitous urban applications evolved from a network of stationary sensor infrastructure to a network of mobile sensor nodes. To foster cities' efficiency and sustainability, the practitioners use sensing infrastructure to collect and analyse data for traffic management, air quality monitoring, noise pollution monitoring, urban planning and healthcare. Most applications collect highly individual and sensitive information like location, time, audio and video. Even with anonymous contributions, a mistrustful application server or an adversary can infer routine and identity and predict a participant's behaviour based on their time and location information. Due to these severe privacy risks, participating users might opt out of urban sensing applications, defeating the entire purpose of large-scale sensing.

Various approaches have been proposed in the literature to address privacy concerns for urban sensing applications [5, 10–12, 14]. The traditional methods either compromise data quality or are too expensive to incite significant participation. *Federated learning* (FL) appears to be a potential approach for urban sensing systems to address these issues. In federated learning, local model update rather than participants' sensitive data is transmitted to a central server for aggregation to generate a global model. Although federated learning enables distributed training, the participants' data may still be at risk of being leaked due to inference attacks. To prevent such information leakage, *secure multi-party computation* (SMC) securely aggregates the local model updates of several participants before transmitting them to the application server.

In their recent work, Bonawitz et al. [4] proposed a secure aggregation protocol based on cryptographic primitives with quadratic computation cost for participants. Further, to guarantee the correctness of aggregation, Guo et al. proposed *VeriFL* [7] involving cryptographic primitives of secret sharing, in which a trusted centralized server is assigned the task of generating key pairs. Bell et al. [2] presented a secure aggregation scheme that achieves poly-logarithmic communication and computation per participant. Fereidooni et al. presented *SAFELearn* [6], a secure aggregation design adaptive to security and efficiency requirements and involves two communication rounds. A secure aggregation protocol *Turbo-Aggregate* proposed by So et al. [13] employs a multi-group circular strategy. The scheme leverages additive secret sharing and adds aggregation redundancy via Lagrange coding to enable robustness against dropped or delayed participants. It involves one communication round per participant and can tolerate a dropout rate of up to 50%.

For real time FL-based urban sensing systems, the application model needs to be updated frequently. To achieve this, data is expected to be recorded periodically, every few minutes or hours, mostly resulting in data-intensive spatial-temporal model updates. Moreover, the participants may have limited resources, such as poor network connection, processing, and battery resources. Applying the existing SMC schemes as such will result in high space and time complexity and make it resource-intensive for the participants. In this paper, we propose an adaptive SMC protocol for FL-based urban sensing systems that considers the resource constraints of participants. The proposed protocol is called *Resource Adaptive*

*Corresponding author.

(ReAd) Turbo-Aggregate, a modified version of Turbo-Aggregate protocol that securely aggregates data in a single round and can handle up to 50% dropouts. As dropouts are a more prevalent concern in urban sensing applications, we found Turbo Aggregate more suitable. The proposed scheme is designed specifically for handling varying resources of participants in urban sensing applications while retaining the features and advantages of the original protocol. We conduct simulations for urban noise mapping on a synthetic dataset to verify the proposed method. The results show that the proposed scheme gives an acceptable application model while preserving privacy at low space and time complexity for participants compared to the conventional SMC schemes.

The main contribution of this paper is that it develops a resource adaptive variant of the state-of-the-art SMC protocol Turbo Aggregate for inference attack mitigation in FL. The proposed approach, ReAd Turbo Aggregate, is specifically designed for urban sensing applications where the application model is a spatiotemporal map over the area of interest and needs to be periodically updated. ReAd Turbo Aggregate adaptively changes the spatial and temporal granularity of the application model at each update to match participating users' computation and communication resources. The proposed approach is verified on the urban sensing task of ambient noise pollution mapping using simulated data. The rest of the paper is organized as follows: Section 2 describes the overview of the baseline protocol, the proposed ReAd Turbo Aggregate protocol, and the implementation details. Numerical experiments are performed in Section 3, and the results are illustrated in Section 4, followed by conclusion in Section 5.

2 PROPOSED PROTOCOL: READ TURBO AGGREGATE

This section reviews the state-of-the-art secure aggregation protocol: Turbo-Aggregate, proposed by So et al. [13], as our baseline. Furthermore, we present our proposed protocol, specifically tailored for urban sensing applications, which incorporates all the fundamental characteristics of the baseline protocol.

2.1 Overview of Turbo Aggregate Protocol

The protocol employs a multi-group circular aggregation strategy. N participating users are randomly divided into L groups of k participants, as shown in Figure 1 such that $N = L \times k$. Contributions are aggregated sequentially in a circular fashion starting from first group till the last group. The protocol leverages additive masking to enable privacy and lagrange coding to effectively tolerate dropout up to 50%. Communication between participants via the server is encrypted with a shared secret using Diffie-Hellman key exchange.

We represent the set of participants in a group l who complete the protocol as $v_l \in [N_l]$ while the participants who drop out in the middle are represented as $D_l = [N_l]/v_l$. The objective is to evaluate $\mathcal{M}$ where $\boldsymbol{\varkappa}_i^{(l)}$ is the individual contribution of the participant as in Eq. (1).

$$\mathcal{M} = \sum_{l \in [L]} \sum_{i \in v_l} \boldsymbol{\varkappa}_i^{(l)} \tag{1}$$

Each user preserves its privacy with two types of masks which it adds to $\boldsymbol{\varkappa}_i^{(l)}$. It adds a random mask $\boldsymbol{\nu}_i^{(l)}$ that it receives from

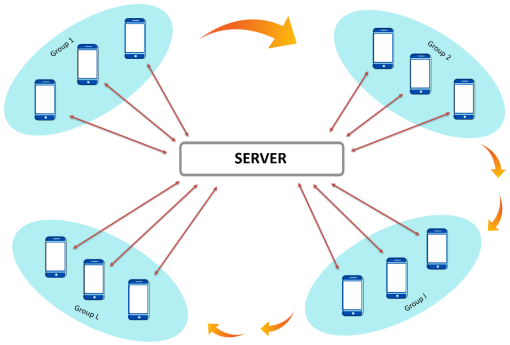


Figure 1: Turbo-Aggregate protocol with L groups

the server. This mask protects data from any subset of colluding co-participants. While sharing their contribution, each participant also employs additive masking to protect data in case the server colludes with a group of participants. Participant i in group l creates N_{l+1} additive masks $\boldsymbol{\rho}_{i,j}^{(l)}$ such that $\sum_{j \in [N_{l+1}]} \boldsymbol{\rho}_{i,j}^{(l)} = 0$ and create its contribution as in Eq. (2). In its turn, participant i shares $\widetilde{\boldsymbol{\varkappa}}_{i,j}^{(l)}$ to every fellow participant j in the next group after double masking.

$$\widetilde{\boldsymbol{\varkappa}}_{i,j}^{(l)} = \boldsymbol{\varkappa}_i^{(l)} + \boldsymbol{\nu}_i^{(l)} + \boldsymbol{\rho}_{i,j}^{(l)} \tag{2}$$

Every participant also evaluates a partial aggregate $\widetilde{\varsigma}_i^{(l)}$ which is also shared with fellow participants in the next group along with $\widetilde{\boldsymbol{\varkappa}}_{i,j}^{(l)}$ in its turn. The partial aggregate is calculated from the values $(\{\widetilde{\boldsymbol{\varkappa}}_{h,i}^{(l-1)}, \widetilde{\varsigma}_h^{(l-1)}\})$ current participant i received from active members in the previous group $(l-1)$ via a recursive relation defined in Eq. (3). Participants can reconstruct the values for the participants in the previous group that dropped out with Lagrange coding [13].

$$\widetilde{\varsigma}_i^{(l)} = \frac{\sum_{j \in [N_{l-1}]} \widetilde{\varsigma}_j^{(l-1)}}{N_{l-1}} + \sum_{j \in [v_{l-1}]} \widetilde{\boldsymbol{\varkappa}}_{j,i}^{(l-1)} \tag{3}$$

The beauty of additive masking lies in the fact that the average of the aggregate values $\widetilde{\varsigma}_i^{(l)}$ from all participants in a group l evaluates to the sum of local models of surviving participants till group $l-1$ masked by the random masks from the server.

$$\varsigma^{(l+1)} = \frac{\sum_{i \in [N_l]} \widetilde{\varsigma}_i^{(l)}}{N_l} \tag{4}$$

$$\varsigma^{(l+1)} = \sum_{m \in [l-1]} \sum_{j \in v_m} \boldsymbol{\varkappa}_j^{(m)} + \sum_{m \in [l-1]} \sum_{j \in v_m} \boldsymbol{\nu}_j^{(m)} \tag{5}$$

A random group $(final)$ of k available participants receive the last set of values $\{\widetilde{\boldsymbol{\varkappa}}_{i,j}^{(l)}, \widetilde{\varsigma}_i^{(l)}\}$ where $i \in [N_L]$ and $j \in [N_{final}]$. Participants j in N_{final} evaluate $\widetilde{\varsigma}_j^{(final)}$ as per Eq. (6) and share it with the server.

$$\widetilde{\varsigma_j^{(final)}} = \frac{\sum_{i \in [N_L]} \widetilde{\varsigma}_i^{(L)}}{N_L} + \sum_{i \in [v_L]} \widetilde{\boldsymbol{\varkappa}}_{i,j}^{(L)} \tag{6}$$

The server computes the average of partial aggregates received from the k participants in the $final$ group using Eq. (7), where additive masks cancel out. Further, subtraction of the random masks of each surviving participant from $\varsigma^{(server)}$ evaluates to $\mathcal{M}$.

$$\varsigma^{(server)} = \frac{\sum_{i \in [N_{final}]} \widetilde{\varsigma}_i^{(final)}}{N_{final}} \tag{7}$$

The partial aggregate $\varsigma_i^{(l)}$ or the masked contributions $\widetilde{\varkappa}_{i,j}^{(l)}$ at any stage do not reveal any information about the underlying data. In a special and unlikely scenario, participants in a group $l \in [L]$ can collude with the server and attempt to retrieve the aggregate of participants' contributions until Group-$l-1$. However, it would be impossible for them to identify and trace the individual contributions that form the aggregate. Thus the protocol is secure and improves privacy for participants in all practical cases.

2.2 ReAd Turbo Aggregate Protocol

We propose Resource Adaptive (ReAd) Turbo-Aggregate protocol which aptly modifies the secure multi-party aggregation scheme Turbo-Aggregate [13] for urban sensing applications. The topology for the ReAd Turbo Aggregate protocol follows the parent protocol, which employs a multi-group circular aggregation strategy. ReAd Turbo Aggregate protocol inherits the privacy of Turbo Aggregate as it leverages the additive masking and distribution of contribution as in Turbo Aggregate protocol which grants it security. Similar to Turbo Aggregate, the proposed protocol effectively tolerates up to 50% user dropout via Lagrange coding. ReAd Turbo Aggregate protocol introduces federated learning for urban sensing applications and provides the opportunity to make it adaptive to suit the battery and network resources of participants.

In urban sensing applications, the global model is usually a spatio-temporal representation of parameters like ambient noise or pollutants as a function of location and time. The geographical region of interest is divided into a (r, c) size grid of small cells of fixed length and width. The grid is correspondingly represented by a $r \times c$ sized matrix R, the global model for mapping the parameter. In the first round of aggregation, R is a null matrix passed to the participants in the first group from the server at the start of the protocol. Each participant successively adds their matrix contribution $\varkappa$ along with additive masking as per the Turbo Aggregate protocol. For a participant spatially located in a particular grid cell (i, j) with scalar observation α, the matrix contribution $\varkappa$ would be defined as in Eq. (8).

$$\varkappa(l, m) = \begin{cases} \alpha & \text{if } (l, m) = (i, j) \\ 0 & \text{if } elsewhere \end{cases} \tag{8}$$

Thus, each participant makes its local contribution to the corresponding cell in global model R as shown in Figure 2. At the end of the protocol, we achieve the required aggregate R where each cell $R(i, j)$ is the sum of scalar observation for the participants located in the grid cell (i, j).

The global model in urban sensing is a smooth continuous surface of the physical parameter being modelled, and the values for neighboring cells would be very similar due to the slowly varying property of the natural phenomenon being observed. Most urban sensing tasks require real-time sensing, which in turn, results in

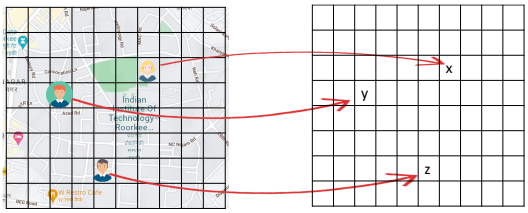


Figure 2: Spatial region with grid for representation as model with (x,y,z) as scalar observations from different participants

frequent updates from participants. Practitioners also desire a granular and precise model with finer resolution in the area of interest. Applying the FL scheme using SMC will provide security to the participants, but it may be computationally expensive to have frequent updates from a wider pool of participants. In ReAd Turbo-Aggregate, we propose an additional hyperparameter; *grouping parameter* (θ) to manage the complexity of the aggregation scheme while achieving acceptable results [8] at low computation and communication cost to the participants. The parameter $\theta \geq 1$ is a natural number, which is decided by the server after the evaluation of hardware and network constraints of participants and shared with all participants. We define a neighborhood as a $\theta \times \theta$ area and all participants in the same neighborhood contribute their local observations to a cell in the global model. Previously if a participant is spatially located in (i, j) cell, it would have contributed to (i, j) cell in the R. With θ, the participant would now effectively contribute to $\theta \times \theta$ cells in R. Although the global model is a (r, c) matrix, the model in transit referred as H would be of size $(r/\theta, c/\theta)$ to as shown in Figure 3 for an example case of $\theta = 2$. Thus, the model's size in transit is

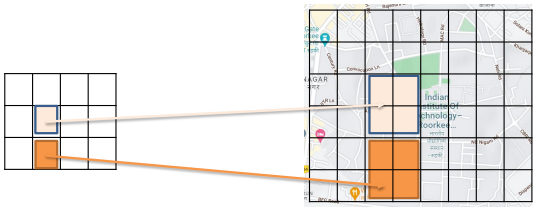


Figure 3: Mapping from model H (*left*) to spatial region (*right*) for $\theta = 2$

reduced by $\theta \times \theta$, reducing the computational complexity. In ReAd Turbo-Aggregate, each participant, spatially located in (i, j) cell with scalar observation α will evaluate its contribution to the model $\varkappa$ as:

$$\varkappa(l, m) = \begin{cases} \alpha & \text{if } (l, m) = (\lfloor i/\theta \rfloor, \lfloor j/\theta \rfloor) \\ 0 & elsewhere \end{cases} \tag{9}$$
$$l \in [1, \lfloor r/\theta \rfloor], m \in [1, \lfloor c/\theta \rfloor]$$

Just as the circular aggregation for H is performed, an aggregation of a *Count* matrix of the same order as that of H is also performed. *Count* facilitates in reconstructing an approximate R from H. $Count(i, j)$ stores the count or number of participants who made contributions at $H(i, j)$. Aggregation for *Count* is done the

Table 1: Time Complexity with n: Number of Participants, k: Members in Each Level, l: Size=$r \times c$, d: Dropout ratio [0-0.5], θ: Grouping Parameter

Protocol	Communication Server	Computation Server	Communication Client	Computation Client
Turbo-Aggregate	$O(nkl)$	$O(kl)$	$O(kl)$	$O(\alpha lk + \beta lk^2 + \gamma lk^2 d)$
ReAd Turbo-Aggregate	$O(nkl/\theta^2)$	$O(kl/\theta^2)$	$O(kl/\theta^2)$	$O(\alpha lk/\theta^2 + \beta lk^2/\theta^2 + \gamma lk^2 d/\theta^2)$

same way as H, sequentially or in parallel. Each available participant's matrix contribution y to $Count$ will be:

$$y(l,m) = \begin{cases} 1 & \text{if } \varkappa(l,m) \neq 0 \\ 0 & elsewhere \end{cases} \\ where, l \in [1, \lfloor r/\theta \rfloor], m \in [1, \lfloor c/\theta \rfloor] \tag{10}$$

Once the server receives H and $Count$, it evaluates an approximate R. Each value $Count(i,j)$ and $H(i,j)$ is used to fill values in the $\theta \times \theta$ neighborhood it represented with averaging function. Each cell in R is evaluated as:

$$R(l,m) = \begin{cases} \frac{H(\lfloor l/\theta \rfloor, \lfloor m/\theta \rfloor)}{Count(\lfloor l/\theta \rfloor, \lfloor m/\theta \rfloor)} & \text{if } Count(\lfloor l/\theta \rfloor, \lfloor m/\theta \rfloor) \neq 0 \\ 0 & \text{if } Count(\lfloor l/\theta \rfloor, \lfloor m/\theta \rfloor) = 0 \end{cases} \\ where, \quad l \in [1,r], m \in [1,c] \tag{11}$$

The comparison of communication and computational time complexities of Turbo-Aggregate and ReAd Turbo-Aggregate is shown in Table 1. The ReAd Turbo-Aggregate protocol slashes the computation and communication requirements by a factor of θ^2 as the model size is reduced. The tradeoff: the results obtained with $\theta > 1$ are the approximate versions obtained using Turbo-Aggregate, as some information is lost during grouping. The protocol is as secure as the parent protocol as the server performs additional steps associated with grouping parameters before and after aggregation of H and $Count$. Any third-party adversary, curious participant or server cannot infer participants' private information as in between aggregation; the partial aggregates are masked by virtue of additive secret sharing in Turbo Aggregate. After the masks cancel at the end of the protocol, no party can trace the source of contribution in aggregate. Thus, the ReAd Turbo Aggregate protocol is equally secure as the parent Turbo Aggregate protocol.

3 NUMERICAL EXPERIMENTS

We validate two of the hypotheses (listed below) of the proposed ReAd Turbo-Aggregate protocol by applying it to the noise mapping application:

(1) **Hypothesis 1: Application Model Accuracy**
The application model obtained using the ReAd Turbo Aggregate approach with $\theta = 1$ is the same as the expected results from Turbo Aggregate or conventional aggregation without SMC.
(2) **Hypothesis 2: Faster but Approximate Model**
As we increase θ, application model results from the proposed ReAd Turbo Aggregate protocol are close to the output of Turbo Aggregate or conventional aggregation without SMC but at significantly lower time complexity.

The protocol is applied to simulated data obtained from the noise-planet project [1]. The synthetic data is created with the NoiseModelling GIS Tool [3], which generates environmental noise maps for urban regions based on building, land use, transportation network, and traffic data. In this simulated experiment, we choose a $1km \times 1.5km$ region from Lorient in northwestern France, as illustrated in Figure 4. For ten days (day, evening, and night), a

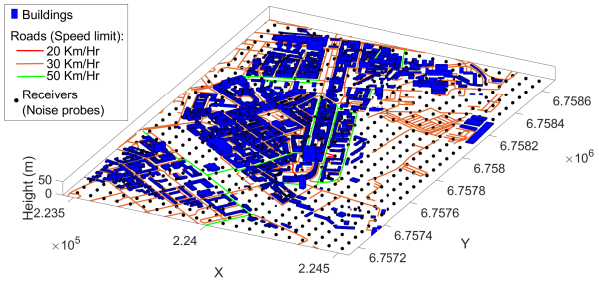


Figure 4: Simulation region consisting of buildings, roads, and receiver locations

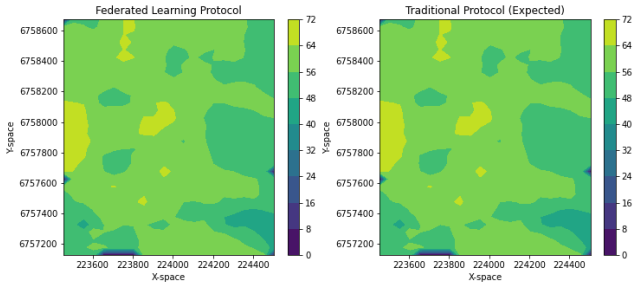
set of noise probes (receivers) used to replicate dynamic noise measurements create a dataset of 30 noise maps with a number of participating users between 150 and 1500 [9]. Participants record noise levels as they move along the region of interest, mapped to a rectangular grid with each square cell of side 50 units. Thus, the global model $Model$ covering the entire region of interest is a matrix of size 32×22. In case sufficient participants are available, each cell in the model shall have a non-zero value, and we obtain a noise map over the area of interest. In most practical cases, the density of participants will vary with location, and we may obtain multiple zero valued cells after aggregation. Approximate values for these cells are generated using the *inverse distance weighing* (IDW) spatial interpolation with $d_{max} = 100$ (region of influence) and $q = 2$ (strength of influence) as hyperparameters. For these experiments, we performed all operations in a field of large prime $p = 1000000007$ (selected arbitrarily).

In traditional aggregation simulation, each participating user shares its location and observation (measurement of the physical phenomenon) as a tuple (x, y, z) with the server, and aggregation is performed sequentially. Once the server receives the tuple data from a user, it updates the global model by applying IDW interpolation. In the ReAd Turbo Aggregate simulation, the server is aware of users who are available to participate in the protocol before the start of aggregation. The server randomly partitions them into groups of size $k = 4$ and aggregates H and $Count$ variables while

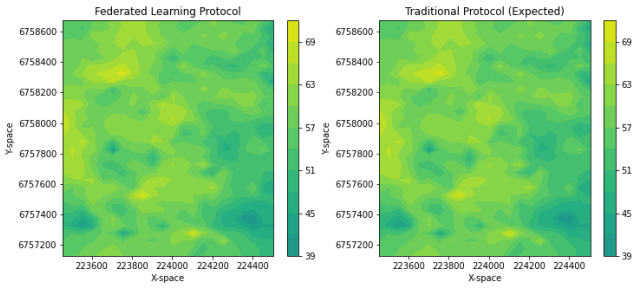
managing any random dropouts as per Turbo Aggregate protocol. Post aggregation, the server evaluates R and applies IDW interpolation to generate a smooth noise map surface. In experiments, the number of participants (N) varies between 150 and 1500.

4 RESULTS

We plot the results after aggregation via FL protocol using ReAd Turbo Aggregate and traditional aggregation (centralized approach) for varying timestamps and the number of participants. We observe that with no grouping ($\theta = 1$), we get exactly the same noise surface using both the aggregation schemes, proving hypothesis 1, i.e., the application model accuracy of the ReAd Turbo Aggregate protocol. In Figure 5, one can observe that the contour plots overlap in both protocols, even with many participants.



(a) t = 1 with n ~ 150



(b) t = 30 with n ~ 1500

Figure 5: Noise surface output in federated learning using ReAd Turbo Aggregate vs traditional learning for $\theta = 1, k = 4, DropoutRatio = 0, d_{max} = 100$

To prove hypothesis 2 concerning reduced computational complexity to generate an approximate application model, we experiment with different values of θ. Keeping other hyperparameters (k, $DropoutRatio$, d_{max}) constant, we observe that complexity reduces sharply as θ increases for a large range of number of participants as shown in Figure 6. Table 2 lists the relative average processing time for each participant for $\theta = \{1, 2, 3\}$ evaluated from the slope of curves corresponding to θ in Figure 6. The computations for ReAd Turbo-Aggregate with $\theta = 2$ and $\theta = 3$ are approximately 3.9 and 9.6 times faster than those for native Turbo-Aggregate ($\theta = 1$). When $\theta \neq 1$, the generated noise surface (application model) is an approximate or blurred version of the one generated from traditional aggregation. We vary the parameters θ and d_{max} and plot the noise map obtained for various configurations in Figure 7. View (a) shows

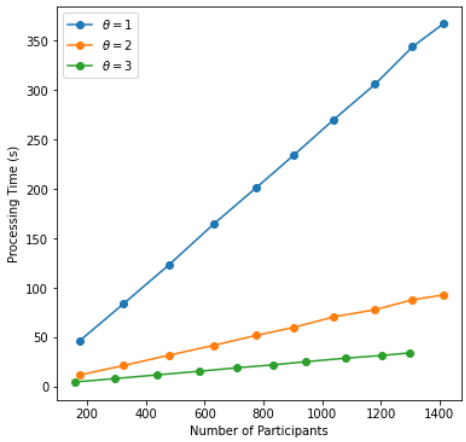


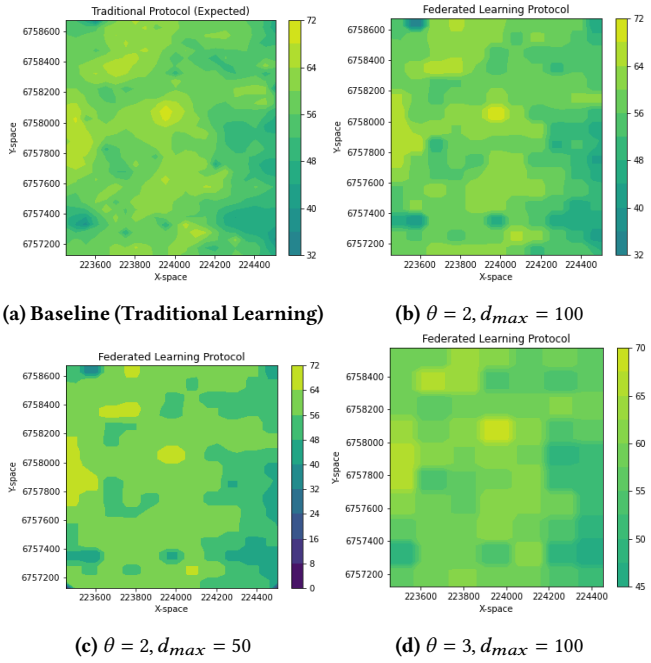
Figure 6: Processing time versus number of participants (N) for different values of θ ($k = 4, DropoutRatio = 0, d_{max} = 100$)

Table 2: Processing time per participant versus θ to determine the factor by which it is influenced

θ	**Processing time for each participant**	**Processing with respect to $\theta = 1$**
1	0.26	1x
2	0.0667	3.9x faster
3	0.027	9.6x faster

the ground truth noise map while views (b), (c), and (d) show the interpolated noise map surface obtained by ReAd Turbo-Aggregate for $\theta = 2, d_{max} = 100$, $\theta = 2, d_{max} = 50$, and $\theta = 3, d_{max} = 100$ respectively. The noise map generated using $\theta = 2$, and $d_{max} = 100$ (view (b)) seems to be the least distorted, followed by one generated using $\theta = 2$, and $d_{max} = 50$ (Figure 7(c)). The noise surface generated using $\theta = 3$ is highly distorted and has square-shaped contours because a single value is extrapolated to 3×3 region.

To quantify the error in the application model (noise map), we evaluate the *root mean square error* (RMSE) between the baseline noise map and those generated using ReAd Turbo-Aggregate for varying values of parameters θ and d_{max}, which is plotted in Figure 8. For fewer participants (≤ 500), RMSE is relatively high due to IDW approximation errors; however, it falls sharply as the number of participants increases. When user density is high, most cells will have non-zero values and will not require IDW interpolation. We also observe that for the same θ but different d_{max} in a high user density environment, RMSE does not vary much. Among all combinations, RMSE for $\theta = 2$, and $d_{max} = 100$ is lowest, while it is maximum for $\theta = \{3, 4\}$, and $d_{max} = 50$. Observations for fewer participants are a good representation of a granular map with sparse user density. Administrators can carefully regulate the parameters in each round with consideration of the density of users, feasible space and time complexity in aggregation and acceptable error in results.



(a) Baseline (Traditional Learning) **(b)** $\theta = 2, d_{max} = 100$

(c) $\theta = 2, d_{max} = 50$ **(d)** $\theta = 3, d_{max} = 100$

Figure 7: Noise map generated using traditional protocol vs federated learning using ReAd Turbo-Aggregate for different hyper parameter settings

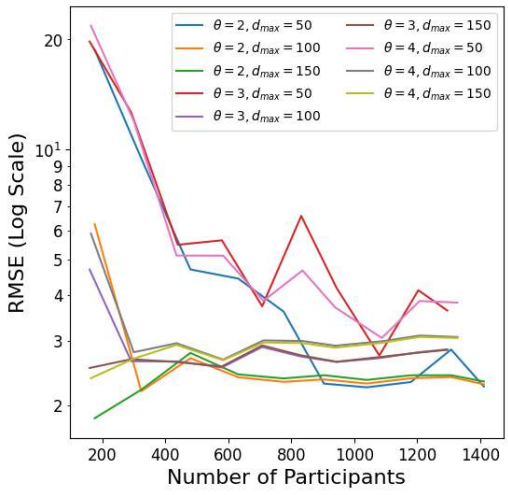


Figure 8: Root mean square error (RMSE) for various θ and d_{max}

5 CONCLUSION

The paper proposes and experimentally demonstrates the Resource Adaptive Turbo-Aggregate protocol for securely aggregating the local model updated from participating users adaptively for federated learning architecture. ReAd Turbo-Aggregate introduces a grouping parameter θ to the Turbo-Aggregate protocol, which can be configured to reduce time complexity for resource-constrained users while providing satisfactory results. The proposed protocol was tested on a synthetic dataset for the noise mapping application. We observe a tradeoff between time complexity and accuracy of the generated noise map. The application can tune the hyper-parameter θ, d_{max}, and k to adaptively select the configuration per the smartphone resources. Future work involves exploring the integration of incremental learning and federated learning techniques to address the challenge of limited storage capacity on clients' devices.

REFERENCES

[1] [n. d.]. Noise-Planet: Scientific tools for environmental noise assessment. http://noise-planet.org/index.html. Accessed: 2020-03-14.

[2] James Henry Bell, Kallista A Bonawitz, Adrià Gascón, Tancrède Lepoint, and Mariana Raykova. 2020. Secure single-server aggregation with (poly) logarithmic overhead. In *ACM SIGSAC Conference on Computer and Communications Security*. 1253–1269.

[3] Erwan Bocher, Gwenaël Guillaume, Judicaël Picaut, Gwendall Petit, and Nicolas Fortin. 2019. NoiseModelling: An Open Source GIS Based Tool to Produce Environmental Noise Maps. *ISPRS International Journal of Geo-Information* 8, 3 (2019), 130.

[4] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2017. Practical secure aggregation for privacy-preserving machine learning. In *ACM SIGSAC Conference on Computer and Communications Security*. 1175–1191.

[5] Jianwei Chen, Huadong Ma, and Dong Zhao. 2017. Private data aggregation with integrity assurance and fault tolerance for mobile crowd-sensing. *Wireless Networks* 23, 1 (2017), 131–144.

[6] Hossein Fereidooni, Samuel Marchal, Markus Miettinen, Azalia Mirhoseini, Helen Möllering, Thien Duc Nguyen, Phillip Rieger, Ahmad-Reza Sadeghi, Thomas Schneider, Hossein Yalame, et al. 2021. SAFELearn: secure aggregation for private federated learning. In *2021 IEEE Security and Privacy Workshops (SPW)*. IEEE, 56–62.

[7] Xiaojie Guo, Zheli Liu, Jin Li, Jiqiang Gao, Boyu Hou, Changyu Dong, and Thar Baker. 2020. Verifl: Communication-efficient and fast verifiable aggregation for federated learning. *IEEE Transactions on Information Forensics and Security* 16 (2020), 1736–1751.

[8] Sparsh Gupta, Ayshika Kapoor, and Dheeraj Kumar. 2023. A Resource Adaptive Secure Aggregation Protocol for Federated Learning based Urban Sensing Systems. In *Joint International Conference on Data Science & Management of Data*. 135–135.

[9] Dheeraj Kumar. 2022. FL-NoiseMap: A Federated Learning-based privacy-preserving Urban Noise-Pollution Measurement System. *Noise Mapping* 9, 1 (2022), 128–145.

[10] Rim Ben Messaoud, Nouha Sghaier, Mohamed Ali Moussa, and Yacine Ghamri-Doudane. 2018. Privacy preserving utility-aware mechanism for data uploading phase in participatory sensing. *IEEE Transactions on Mobile Computing* 18, 9 (2018), 2160–2173.

[11] Chenglin Miao, Wenjun Jiang, Lu Su, Yaliang Li, Suxin Guo, Zhan Qin, Houping Xiao, Jing Gao, and Kui Ren. 2019. Privacy-preserving truth discovery in crowd sensing systems. *ACM Transactions on Sensor Networks (TOSN)* 15, 1 (2019), 1–32.

[12] Kun Niu, Changgen Peng, Weijie Tan, Zhou Zhou, and Yi Xu. 2021. Verifiable Location-Encrypted Spatial Aggregation Computing for Mobile Crowd Sensing. *Security and Communication Networks* 2021 (2021).

[13] Jinhyun So, Başak Güler, and A Salman Avestimehr. 2021. Turbo-aggregate: Breaking the quadratic aggregation barrier in secure federated learning. *IEEE Journal on Selected Areas in Information Theory* 2, 1 (2021), 479–489.

[14] Gaoqiang Zhuo, Qi Jia, Linke Guo, Ming Li, and Pan Li. 2016. Privacy-preserving verifiable data aggregation and analysis for cloud-assisted mobile crowdsourcing. In *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*. IEEE, 1–9.