ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) | 978-1-7281-6327-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICASSP49357.2023.10095450

CODED MATRIX COMPUTATIONS FOR D2D-ENABLED LINEARIZED FEDERATED LEARNING

Anindya Bijoy Das[†] *Aditya Ramamoorthy*[⋆] *David J. Love*[†] *Christopher G. Brinton*[†]

[†]School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907 USA
[⋆]Department of Electrical and Computer Engineering, Iowa State University, Ames, IA 50010 USA

ABSTRACT

Federated learning (FL) is a popular technique for training a global model on data distributed across client devices. Like other distributed training techniques, FL is susceptible to straggler (slower or failed) clients. Recent work has proposed to address this through device-to-device (D2D) offloading, which introduces privacy concerns. In this paper, we propose a novel straggler-optimal approach for coded matrix computations which can significantly reduce the communication delay and privacy issues introduced from D2D data transmissions in FL. Moreover, our proposed approach leads to a considerable improvement of the local computation speed when the generated data matrix is sparse. Numerical evaluations confirm the superiority of our proposed method over baseline approaches.

***Index Terms*—** Distributed Computing, Federated Learning, Stragglers, Heterogeneous Edge Computing, Privacy.

1. INTRODUCTION

Contemporary computing platforms are hard-pressed to support the growing demands for AI/ML model training at the network edge. While advances in hardware serve as part of the solution, the increasing complexity of data tasks and volumes of data will continue impeding scalability. In this regard, federated learning (FL) has become a popular technique for training machine learning models in a distributed manner [1–3]. In FL, the edge devices carry out the local computations, and the server collects, aggregates and updates the global model.

Recent approaches have looked at linearizing the training operations in FL [1, 4]. This is advantageous as it opens the possibility for coded matrix computing techniques that can improve operating efficiency. Specifically, in distributed settings like FL, the overall job execution time is often dominated by slower (or failed) worker nodes, which are referred to as stragglers. Recently, a number of coding theory techniques [5–14] have been proposed to mitigate stragglers in distributed matrix multiplications. A toy example [5] of such a technique for computing $\mathbf{A}^T\mathbf{x}$ across three clients is to partition $\mathbf{A}$ as $\mathbf{A} = [\mathbf{A}_0 \mid \mathbf{A}_1]$, and to assign them the job of computing $\mathbf{A}_0^T\mathbf{x}$, $\mathbf{A}_1^T\mathbf{x}$ and $(\mathbf{A}_0 + \mathbf{A}_1)^T\mathbf{x}$, respectively. In a linearized FL setting, $\mathbf{A} \in \mathbb{R}^{t\times r}$ is the data matrix and $\mathbf{x} \in \mathbb{R}^t$ is the model parameter vector. While each client has half of the total computational load, the server can recover $\mathbf{A}^T\mathbf{x}$ if *any* two clients return their results, i.e., the system is resilient to one straggler. If each of n clients computes $1/k_A$ fraction of the whole job of computing $\mathbf{A}^T\mathbf{x}$, the number of stragglers that the system can be resilient to is upper bounded by $n - k_A$ [7].

In contemporary edge computing systems, task offloading via device-to-device (D2D) communications has also been proposed for straggler mitigation. D2D-enabled FL has recently been studied [2, 15, 16], but can add considerable communication overhead as well as compromise data privacy. In this work, we exploit matrix coding in linearized FL to mitigate these challenges. Our straggler-optimal matrix computation scheme reduces the communication delay significantly compared to the techniques in [7, 9, 12]. Moreover, unlike [7, 9, 12, 13, 17], our scheme allows a client to access a limited fraction of matrix $\mathbf{A}$, and provides a considerable protection against information leakage. In addition, our scheme is specifically suited to sparse matrices with a significant gain in computation speed.

2. NETWORK AND LEARNING ARCHITECTURE

We consider a D2D-enabled FL architecture consisting of $n = k_A + s$ clients, denoted as W_i for $i = 0, 1, \ldots, n-1$. The first k_A of them are active clients (responsible for both data generation and local computation) and the next $s < k_A$ are passive clients (responsible for local computation only).

Assume that the i-th device has local data $(\mathbf{D}_i, \mathbf{y}_i)$, where $\mathbf{D}_i$ and $\mathbf{y}_i$ are the block-rows of full system dataset $(\mathbf{D}, \mathbf{y})$. Under a linear regression-based ML model, the global loss function is quadratic, i.e., $f(\beta_\ell) = ||\mathbf{D}\beta_\ell - \mathbf{y}||^2$, where the model parameter after iteration ℓ is obtained through gradient methods as $\beta_\ell = \beta_{\ell-1} - \mu_\ell \nabla_\beta f(\beta_{\ell-1})$ and μ_ℓ is the stepsize. Based on the form of $\nabla_\beta f(\beta_\ell)$, the FL local model update at each device includes multiplying the local data matrix $\mathbf{D}_i$ with parameter β_ℓ. For this reason, recent work has also investigated linearizing non-linear models for FL by leveraging kernel embedding techniques [1]. Thus, our aim is to compute $\mathbf{A}^T\mathbf{x}$ – an arbitrary matrix operation during FL training – in a distributed fashion such that the system is resilient to s stragglers. Our assumption is that any active client W_i generates a block-column of matrix $\mathbf{A}$, denoted as $\mathbf{A}_i$, $i = 0, 1, \ldots, k_A - 1$, such that

$$\mathbf{A} = \begin{bmatrix}\mathbf{A}_0 & \mathbf{A}_1 & \ldots & \mathbf{A}_{k_A-1}\end{bmatrix}. \tag{1}$$

Algorithm 1: Proposed scheme for distributed matrix-vector multiplication

```
Input  : Matrix $\mathbf{A}_i$ generated in active client $i$ for
         $i = 0, 1, \dots, k_A - 1$, vector $\mathbf{x}$, total $n$
         clients including $s < k_A$ passive clients.
1  Set weight $\omega_A = s + 1$ ;
2  Denote client $i$ as $W_i$, for $i = 0, 1, \dots, n-1$;
3  for $i \leftarrow 0$ to $k_A - 1$ do
4      Define $T_i = \{i+1, \dots, i + \omega_A - 1\}$ (mod $k_A$);
5      Send $\mathbf{A}_j$, where $j \in T_i$, from $W_j$ to $W_i$;
6      Client $W_i$ creates a random vector $\mathbf{r}$ of length $k_A$,
         computes $\tilde{\mathbf{A}}_i = \sum_{q \in T_i} r_q \mathbf{A}_q$ and $\tilde{\mathbf{A}}_i^T \mathbf{x}$;
7  end
8  for $i \leftarrow 0$ to $s - 1$ do
9      $W_i$ creates random vector $\tilde{\mathbf{r}}$ of size $k_A$, computes
         $\tilde{\mathbf{A}}_{k_A+i} = \sum_{q \in T_i} \tilde{r}_q \mathbf{A}_q$ and sends to $W_{k_A+i}$;
10     Client $W_{k_A+i}$ computes $\tilde{\mathbf{A}}_{k_A+i}^T \mathbf{x}$;
11 end
Output : The server recovers $\mathbf{A}^T\mathbf{x}$ from the returned
         results by the fastest $k_A$ clients.
```

In our approach, every client is responsible to compute the product of a coded submatrix (linear combinations of some block-columns of $\mathbf{A}$) and the vector $\mathbf{x}$. Stragglers will arise in practice from computing speed variations or failures experienced by the clients at particular times [8,17,18]. Now, similar to [15,16,19], we assume that there is a set of trusted neighbor clients for every device to transmit its data via D2D communications. The passive clients receive coded submatrices only from active clients. Unlike the approaches in [1,3,4,20], we assume that the server cannot access to any uncoded/coded local data generated in the edge devices and is only responsible for transmission of vector $\mathbf{x}$ and for decoding $\mathbf{A}^T\mathbf{x}$ once the fastest clients return the computed submatrix-vector products.

3. HOMOGENEOUS EDGE COMPUTING

Here we assume that each active client generates equal number of columns of $\mathbf{A}$ (i.e. all $\mathbf{A}_i$'s have the same size in (1)) and all the clients are rated with the same computation speed. In this scenario, we propose a distributed matrix-vector multiplication scheme in Alg. 1 which is resilient to any s stragglers.

The main idea is that any active client W_j generates $\mathbf{A}_j$, for $0 \leq j \leq k_A - 1$ and sends it to another active client W_i, if $j = i+1, i+2, \dots, i + \omega_A - 1$ (modulo k_A). Here we set $\omega_A = s + 1$, thus, any data matrix $\mathbf{A}_j$ needs to be sent to only $\omega_A - 1 = s$ other clients. Then, active client W_j computes a linear combination of $\mathbf{A}_i, \mathbf{A}_{i+1}, \dots, \mathbf{A}_{i+\omega_A-1}$ (indices modulo k_A) where the coefficients are chosen randomly from a continuous distribution. Next, active client W_i sends another random linear combination of the same submatrices to W_{i+k_A} (a passive client), when $i = 0, 1, \dots, s-1$. Note that all n clients receive the vector $\mathbf{x}$ from the server. Now the job of each client is to compute the product of their respective coded submatrix and the vector $\mathbf{x}$. Once the fastest k_A clients finish and send their computation results to the server, it decodes $\mathbf{A}^T\mathbf{x}$ using the corresponding random coefficients. The following theorem establishes the resiliency of Alg. 1 to stragglers.

Theorem 1. Assume that a system has n clients including k_A active and s passive clients. If we assign the jobs according to Alg. 1, we achieve resilience to any $s = n - k_A$ stragglers.

Proof. In order to recover $\mathbf{A}^T\mathbf{x}$, according to (1), we need to decode all k_A vector unknowns, $\mathbf{A}_0^T\mathbf{x}, \mathbf{A}_1^T\mathbf{x}, \dots, \mathbf{A}_{k_A-1}^T\mathbf{x}$; we denote the set of these unknowns as $\mathcal{B}$. Now we choose an arbitrary set of k_A clients each of which corresponds to an equation in terms of ω_A of those k_A unknowns. Denoting the set of k_A equations as $\mathcal{C}$, we have $|\mathcal{B}| = |\mathcal{C}| = k_A$.

Now we consider a bipartite graph $\mathcal{G} = \mathcal{C} \cup \mathcal{B}$, where any vertex (equation) in $\mathcal{C}$ is connected to some vertices (unknowns) in $\mathcal{B}$ which have participated in the corresponding equation. Thus, each vertex in $\mathcal{C}$ has a neighborhood of cardinality ω_A in $\mathcal{B}$. Our goal is to show that there exists a perfect matching among the vertices of $\mathcal{C}$ and $\mathcal{B}$. We argue this according to Hall's marriage theorem [21] for which we need to show that for any $\bar{\mathcal{C}} \subseteq \mathcal{C}$, the cardinality of the neighbourhood of $\bar{\mathcal{C}}$, denoted as $\mathcal{N}(\bar{\mathcal{C}}) \subseteq \mathcal{B}$, is at least as large as $|\bar{\mathcal{C}}|$. Thus, for $|\bar{\mathcal{C}}| = m \leq k_A$, we need to show that $|\mathcal{N}(\bar{\mathcal{C}})| \geq m$.

Case 1: First we consider the case that $m \leq 2s$. We assume that $m = 2p, 2p-1$ where $1 \leq p \leq s$. Now according to Alg. 1, the participating unknowns are shifted in a cyclic manner among the equations. If we choose any δ clients out of the first k_A clients $(W_0, W_1, W_2, \dots, W_{k_A-1})$, according to the proof of cyclic scheme in Appendix C in [8], the minimum number of total participating unknowns is $\min(\omega_A + \delta - 1, k_A)$, where $\omega_A = s+1$. Now according to Alg. 1, same unknowns participate in two different equations corresponding to two different clients, W_j and W_{k_A+j}, where $j = 0, 1, \dots, s-1$. Thus, for any $|\bar{\mathcal{C}}| = m = 2p, 2p-1 \leq 2s$, we have

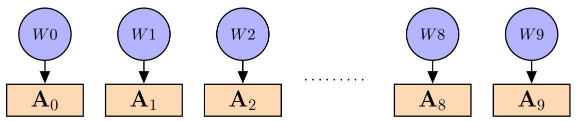
$$\begin{aligned}|\mathcal{N}(\bar{\mathcal{C}})| &\geq \min\left(\omega_A + \lceil m/2 \rceil - 1, k_A\right) \\ &= \min\left(\omega_A + p - 1, k_A\right) = \min\left(s + p, k_A\right) \geq m.\end{aligned}$$

Case 2: Now we consider the case where $m = 2s + q$, $1 \leq q \leq k_A - 2s$. We need to find the minimum number of unknowns which participate in any set of m equations. Now, the same unknowns participate in two different equations corresponding to two different clients, W_j and W_{k_A+j}, where $j = 0, 1, \dots, s-1$. Thus, the additional q equations correspond to at least q additional unknowns until the total number of participating unknowns is k_A. Therefore, in this case

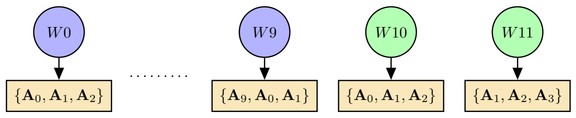
$$\begin{aligned}|\mathcal{N}(\bar{\mathcal{C}})| &\geq \min\left(\omega_A + \lceil 2s/2 \rceil + q - 1, k_A\right) \\ &= \min\left(\omega_A + s + q - 1, k_A\right) = \min\left(2s + q, k_A\right) \geq m.\end{aligned}$$

Thus, for any $m \leq k_A$ (where $|\bar{\mathcal{C}}| = m$), we have shown that $|\mathcal{N}(\bar{\mathcal{C}})| \geq |\bar{\mathcal{C}}|$. So, there exists a perfect matching among the vertices of $\mathcal{C}$ and $\mathcal{B}$ according to Hall's marriage theorem.

Now we consider the largest matching where vertex $c_i \in \mathcal{C}$ is matched to vertex $b_j \in \mathcal{B}$, which indicates that b_j participates in the equation corresponding to c_i. Let us consider a



(a): Data generation among the active clients.


$W0$
$W9$
$W10$
$W11$
$\{\mathbf{A}_0, \mathbf{A}_1, \mathbf{A}_2\}$
$\{\mathbf{A}_9, \mathbf{A}_0, \mathbf{A}_1\}$
$\{\mathbf{A}_0, \mathbf{A}_1, \mathbf{A}_2\}$
$\{\mathbf{A}_1, \mathbf{A}_2, \mathbf{A}_3\}$

(b): Coded submatrix allocation among all the clients.

Fig. 1: (a) Data generation and (b) submatrix allocation for $n = 12$ clients according to Alg. 1 including $k_A = 10$ active and $s = 2$ passive clients. Any $\{\mathbf{A}_j, \mathbf{A}_k, \mathbf{A}_\ell\}$ indicates a random linear combination of the corresponding submatrices. Any W_i obtains a random linear combination of $\mathbf{A}_i, \mathbf{A}_{i+1}$ and $\mathbf{A}_{i+2}$ (indices reduced mod 10).

$k_A \times k_A$ system matrix where row i corresponds to the equation associated to c_i. Now we replace this row i by $\mathbf{e}_j$ which is a unit row-vector of length k_A with j-th entry being 1, and 0 otherwise. Thus we have a $k_A \times k_A$ matrix where each row has only one non-zero entry which is 1. Since we have a perfect matching, this $k_A \times k_A$ matrix has only one non-zero entry in every column. This is a permutation of the identity matrix, and thus, is full rank. Since the matrix is full rank for a choice of definite values, according to Schwartz-Zippel lemma [22], it will be full rank for random choices of non-zero entries. Thus, the server can recover all k_A unknowns from any k_A clients, hence the system is resilient to any $s = n - k_A$ stragglers. ■

Example 1. Consider a homogeneous system of $k_A = 10$ active clients and $s = 2$ passive clients. According to Alg. 1, $\omega_A = s + 1 = 3$, and client W_i $(0 \le i \le 11)$ has a random linear combination of $\mathbf{A}_i, \mathbf{A}_{i+1}$ and $\mathbf{A}_{i+2}$ (indices modulo 10) as shown in Fig. 1. Thus, according to Theorem 1, this system is resilient to $s = 2$ stragglers. Note that our scheme requires any active client to send its local data matrix to only up to $s + 1 = 3$ other clients, thus involves a significantly lower communication cost in comparison to the approaches in [7,9].

Remark 1. In comparison to [7,9,13], our proposed approach is specifically suited to sparse data matrices, i.e., most of the entries of $\mathbf{A}$ are zero. The approaches in [7,9,13] assign dense linear combinations of the submatrices which can destroy the inherent sparsity of $\mathbf{A}$, leading to slower computation speed for the clients. On the other hand, our approach assigns linear combinations of limited number of submatrices which preserve the sparsity up to certain level that leads to faster computation.

4. HETEROGENEOUS EDGE COMPUTING

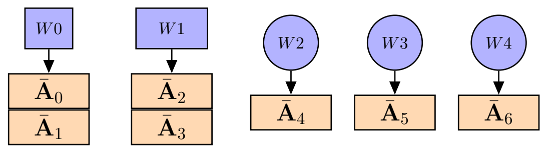
In this section, we extend our approach in Alg. 1 to heterogenous system where the clients may have different data generation capability and different computation speeds. We assume that we have λ different types of devices in the system, with client type $j = 0, 1, \dots, \lambda - 1$. Moreover, we assume that any active client W_i generates $\alpha_i = c_{ij}\alpha$ columns of data matrix $\mathbf{A}$ and any client W_i has a computation speed $\beta_i = c_{ij}\beta$, where W_i is of client type j and $c_{ij} \ge 1$ is an integer. Thus, a higher c_{ij} indicates a "stronger" type client W_i which can process at a c_{ij} times higher computation speed than the "weakest" type device, where α is the number of the assigned columns and β is the number of processed columns per unit time in the "weakest" type device. Note that $\lambda = 1$ and all $c_{ij} = 1$ lead us to the homogeneous system discussed in Sec. 3 where $0 \le i \le n-1$ and $j = 0$.

Now, we have $n = k_A + s$ clients including k_A active and s passive clients in the heterogeneous system. Aligned to the homogeneous system, we assume that the number of passive clients of any type j is less than the number of active clients of the same type. Next, without loss of generality, we sort the indices of active clients in such a way so that, $c_{ij} \ge c_{kj}$ if $i \le k$, for $0 \le i, k \le k_A - 1$. We do the similar sorting for the passive clients too so that $c_{ij} \ge c_{kj}$ if $i \le k$, for $k_A \le i, k \le n - 1$. Now if a client W_i is of client type j, it requires the same time to process $c_{ij} \ge 1$ block-columns (each consisting of α columns) of $\mathbf{A}$ as the "weakest" device to process $c_{ij} = 1$ such block-column. Moreover, if it is an active client, it also generates $\alpha_i = c_{ij}\alpha$ columns of data matrix $\mathbf{A}$. Thus, client W_i can be thought as a collection of c_{ij} homogeneous clients of "weakest" types where each of the active "weakest" clients generates equally α columns of $\mathbf{A}$ and each of the "weakest" clients processes equally α columns.

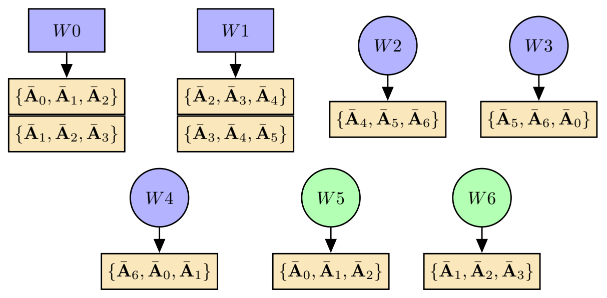
Theorem 2. (a) A heterogeneous system of k_A active and s passive clients of different types can be considered as a homogeneous system of $\bar{k}_A = \sum_{i=0}^{k_A-1} c_{ij}$ active and $\bar{s} = \sum_{i=k_A}^{n-1} c_{ij}$ passive clients of the "weakest" type. Next (b) if the jobs are assigned according to Alg. 1 in the modified homogeneous system of $\bar{n} = \bar{k}_A + \bar{s}$ "weakest" clients, the system can be resilient to $\bar{s}$ such clients.

Proof. Each $\mathbf{A}_k$ (generated in W_k) in (1) is a block-column consisting of $c_{kj}\alpha$ columns of $\mathbf{A}$ when client W_k is of client type j. Thus, for any $k = 0, 1, \dots, k_A - 1$, we can partition $\mathbf{A}_k$ as $\mathbf{A}_k = \begin{bmatrix} \bar{\mathbf{A}}_m & \bar{\mathbf{A}}_{m+1} & \dots & \bar{\mathbf{A}}_{m+c_{kj}-1} \end{bmatrix}$, where $m = \sum_{i=0}^{k-1} c_{ij}$ and each $\bar{\mathbf{A}}_\ell$ is a block-column consisting of α columns of $\mathbf{A}$, $m \le \ell \le m + c_{kj} - 1$. Thus using (1), we can write $\mathbf{A} = \begin{bmatrix} \mathbf{A}_0 & \mathbf{A}_1 & \dots & \mathbf{A}_{\bar{k}_A-1} \end{bmatrix}$, where $\bar{k}_A = \sum_{i=0}^{k_A-1} c_{ij}$. Now from the matrix generation perspective, k_A active clients in a heterogeneous system generating $\bar{k}_A$ block-columns can be considered as the same as $\bar{k}_A$ active clients in a homogeneous system generating *one* block-column each.

Similarly, any client W_i of type j can process $c_{ij}\alpha$ columns in the same time when the "weakest" type device can process α columns. Thus, from the computation speed perspective, k_A active clients and s passive clients in the heterogeneous system can be thought as $\bar{k}_A = \sum_{i=0}^{k_A-1} c_{ij}$ active clients and $\bar{s} = \sum_{i=k_A}^{n-1} c_{ij}$ passive clients, respectively, in a homogeneous system by assigning α coded block-columns to each client. Hence, we are done with the proof of part (a). Moreover, part (b) of the proof is straight-forward from Theorem 1 when we have $\bar{k}_A$ active and $\bar{s}$ passive clients. ■



(a): Data generation among the active clients.



$W0$
$W1$
$W2$
$W3$
$\{\bar{\mathbf{A}}_0, \bar{\mathbf{A}}_1, \bar{\mathbf{A}}_2\}$
$\{\bar{\mathbf{A}}_1, \bar{\mathbf{A}}_2, \bar{\mathbf{A}}_3\}$
$\{\bar{\mathbf{A}}_2, \bar{\mathbf{A}}_3, \bar{\mathbf{A}}_4\}$
$\{\bar{\mathbf{A}}_3, \bar{\mathbf{A}}_4, \bar{\mathbf{A}}_5\}$
$\{\bar{\mathbf{A}}_4, \bar{\mathbf{A}}_5, \bar{\mathbf{A}}_6\}$
$\{\bar{\mathbf{A}}_5, \bar{\mathbf{A}}_6, \bar{\mathbf{A}}_0\}$
$W4$
$W5$
$W6$
$\{\bar{\mathbf{A}}_6, \bar{\mathbf{A}}_0, \bar{\mathbf{A}}_1\}$
$\{\bar{\mathbf{A}}_0, \bar{\mathbf{A}}_1, \bar{\mathbf{A}}_2\}$
$\{\bar{\mathbf{A}}_1, \bar{\mathbf{A}}_2, \bar{\mathbf{A}}_3\}$

(b): Coded submatrix allocation among all the clients.

Fig. 2: A heterogeneous system of $n = 7$ clients where $k_A = 5$ and $s = 2$. (a) Each of W_0 and W_1 generates 2α columns and each of W_2, W_3 and W_4 generates α columns of $\mathbf{A} \in \mathbb{R}^{t \times r}$, where $\alpha = r/7$. (b) Once the jobs are assigned, the system is resilient to stragglers.

Remark 2. The heterogeneous system is resilient to $\bar{s}$ block-column processing. The number of straggler clients that the system is resilient to can vary depending on the client types.

Example 2. Consider the example in Fig. 2 consisting of $n = 7$ clients. There are $k_A = 5$ active clients which are responsible for data matrix generation. Let us assume, W_0 and W_1 are of type 1 clients which generate twice as many columns of $\mathbf{A}$ than W_2, W_3 and W_4 which are of type 0 clients. The jobs are assigned to all clients (including $s = 2$ passive clients) according to Fig. 2(b). It can be verified that this scheme is resilient to *two* type 0 clients or *one* type 1 client.

5. NUMERICAL EVALUATION

In this section, we compare the performance of our proposed approach against different competing methods [7, 9, 13] in terms of different metrics for distributed matrix computations from the federated learning aspect. Note that the approaches in [1, 4] require the edge devices to transmit some coded columns of matrix $\mathbf{A}$ to the server which is not aligned with our assumptions. In addition, the approaches in [8] and [11] do not follow the same network learning architecture as ours. Therefore, we did not include them in our comparison.

Communication Delay: We consider a homogeneous system of $n = 20$ clients each of which is a `t2.small` machine in AWS (Amazon Web Services) Cluster. Here, each of $k_A = 18$ active clients generates $\mathbf{A}_i$ of size 12000×1000, thus the size of $\mathbf{A}$ is 12000×18000. The server sends the parameter vector $\mathbf{x}$ of length 12000 to all 20 clients including $s = 2$ passive clients. Once the preprocessing and computations are carried out according to Alg. 1, the server recovers $\mathbf{A}^T\mathbf{x}$ as soon as it receives results from the fastest $k_A = 18$ clients, thus the system is resilient to any $s = 2$ stragglers.

Table 1: Comparison among different approaches in terms of communication delay for a system with $n = 20$, $k_A = 18$ and $s = 2$.

POLY CODE [7]	ORTHO-POLY [9]	RKRP CODE [13]	CONV. CODE [17]	**PROP. SCH.**
$14.13\,s$	$14.02\,s$	$2.49\,s$	$2.56\,s$	**2.21 s**

Table 2: Per client product computation time where $n = 30$, $k_A = 28$, $s = 2$ and $\zeta = 95\%$, 98% or 99% entries of $\mathbf{A}$ are zero.

METHODS	PRODUCT COMP. TIME (IN MS)		
	$\zeta = 99\%$	$\zeta = 98\%$	$\zeta = 95\%$
POLY CODE [7]	54.7	55.2	53.7
ORTHO-POLY [9]	54.3	54.8	55.2
RKRP CODE [13]	55.1	53.4	53.7
CONV. CODE [17]	56.2	55.8	56.8
PROP. SCHEME	**14.9**	**21.1**	**29.6**

Table 1 shows the comparison of the corresponding communication delays (caused by data matrix transmission) among different approaches. The approaches in [7,9] require all active clients to transmit their generated submatrices to all other edge devices. Thus, they lead to much more communication delay than our proposed method which needs an edge device to transmit data to only up to $s + 1 = 3$ other devices. Note that the methods in [13,17] involve similar amounts of communication delay as ours, however, they have other limitations in terms of privacy and computation time as discussed next.

Privacy: Information leakage is introduced in FL when we consider the transmission of local data matrices to other edge devices. To protect against privacy leakage, any particular client should have access to a limited portion of the whole data matrix. Consider the heterogeneous system in example 2 where the clients are honest but curious. In this scenario, the approaches in [7, 9, 13, 17] would allow clients to access the whole matrix $\mathbf{A}$. In our approach, as shown in Fig. 2, clients W_0 and W_1 only have access to $4/7$-th fraction of $\mathbf{A}$ and clients W_2, W_3 and W_4 have access to $3/7$-th fraction of $\mathbf{A}$. This provides significant protection against privacy leakage.

Product Computation Time for Sparse Matrices: Consider a system with $n = 30$ clients where $k_A = 28$ and $s = 2$. We assume that $\mathbf{A}$ is sparse, where each active client generates a sparse submatrix of size 40000×1125. We consider three different scenarios with three different sparsity levels for $\mathbf{A}$ where randomly chosen 95%, 98% and 99% entries of $\mathbf{A}$ are zero. Now we compare our proposed Alg. 1 against different methods in terms of per client product computation time (the required time for a client to compute its assigned submatrix-vector product) in Table 2. The methods in [7, 9, 13, 17] assign linear combinations of $k_A = 28$ submatrices to the clients. Hence, the inherent sparsity of $\mathbf{A}$ is destroyed in the encoded submatrices. On the other hand, our approach combines only $s + 1 = 3$ submatrices to obtain the coded submatrices. Thus, the clients require a significantly less amount of time to finish the respective tasks in comparison to [7, 9, 13, 17].

6. REFERENCES

[1] Saurav Prakash, Sagar Dhakal, Mustafa Riza Akdeniz, Yair Yona, Shilpa Talwar, Salman Avestimehr, and Nageen Himayat, "Coded computing for low-latency federated learning over wireless edge networks," *IEEE Jour. on Sel. Areas in Comm.*, vol. 39, no. 1, pp. 233–250, 2020.

[2] Su Wang, Seyyedali Hosseinalipour, Maria Gorlatova, Christopher G Brinton, and Mung Chiang, "Uav-assisted online machine learning over multi-tiered networks: A hierarchical nested personalized federated learning approach," *IEEE Trans. on Net. and Serv. Manag.*, 2022.

[3] Jer Shyuan Ng, Wei Yang Bryan Lim, Zehui Xiong, Xianbin Cao, Dusit Niyato, Cyril Leung, and Dong In Kim, "A hierarchical incentive design toward motivating participation in coded federated learning," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 1, pp. 359–375, 2022.

[4] Sagar Dhakal, Saurav Prakash, Yair Yona, Shilpa Talwar, and Nageen Himayat, "Coded federated learning," in *IEEE Globecom Workshop*, 2019, pp. 1–6.

[5] Kangwook Lee, Maximilian Lam, Ramtin Pedarsani, Dimitris Papailiopoulos, and Kannan Ramchandran, "Speeding up distributed machine learning using codes," *IEEE Trans. on Info. Th.*, vol. 64, no. 3, pp. 1514–1529, 2018.

[6] Sanghamitra Dutta, Viveck Cadambe, and Pulkit Grover, "Short-dot: Computing large linear transforms distributedly using coded short dot products," in *Proc. of Adv. in Neur. Inf. Proc. Syst.*, 2016, pp. 2100–2108.

[7] Qian Yu, Mohammad Maddah-Ali, and Salman Avestimehr, "Polynomial codes: an optimal design for high-dimensional coded matrix multiplication," in *Proc. of Adv. in Neur. Inf. Proc. Syst.*, 2017, pp. 4403–4413.

[8] Anindya Bijoy Das and Aditya Ramamoorthy, "Coded sparse matrix computation schemes that leverage partial stragglers," *IEEE Trans. on Info. Th.*, vol. 68, no. 6, pp. 4156–4181, 2022.

[9] M. Fahim and V. R. Cadambe, "Numerically stable polynomially coded computing," *IEEE Trans. on Info. Th.*, vol. 67, no. 5, pp. 2758–2785, 2021.

[10] Rashish Tandon, Qi Lei, Alexandros G Dimakis, and Nikos Karampatziakis, "Gradient coding: Avoiding stragglers in distributed learning," in *Proc. of Intl. Conf. on Mach. Learn.*, 2017, pp. 3368–3376.

[11] Anindya Bijoy Das and Aditya Ramamoorthy, "A unified treatment of partial stragglers and sparse matrices in coded matrix computation," *IEEE Jour. on Sel. Area. in Info. Th.*, vol. 3, no. 2, pp. 241–256, 2022.

[12] Sanghamitra Dutta, Mohammad Fahim, Farzin Haddadpour, Haewon Jeong, Viveck Cadambe, and Pulkit Grover, "On the optimal recovery threshold of coded matrix multiplication," *IEEE Trans. on Info. Th.*, vol. 66, no. 1, pp. 278–301, 2020.

[13] A. M. Subramaniam, A. Heidarzadeh, and K. R. Narayanan, "Random Khatri-Rao-product codes for numerically-stable distributed matrix multiplication," in *Proc. of Annual Conf. on Comm., Control, and Computing (Allerton)*, Sep. 2019, pp. 253–259.

[14] Lev Tauz and Lara Dolecek, "Variable coded batch matrix multiplication," *IEEE Jour. on Sel. Area. in Info. Th.*, vol. 3, no. 2, pp. 306–320, 2022.

[15] Su Wang, Mengyuan Lee, Seyyedali Hosseinalipour, Roberto Morabito, Mung Chiang, and Christopher G Brinton, "Device sampling for heterogeneous federated learning: Theory, algorithms, and implementation," in *Proc. of Intl. Conf. on Comp. Comm.*, 2021, pp. 1–10.

[16] Yuwei Tu, Yichen Ruan, Satyavrat Wagle, Christopher G Brinton, and Carlee Joe-Wong, "Network-aware optimization of distributed learning for fog computing," in *Proc. of Intl. Conf. on Comp. Comm.*, 2020, pp. 2509–2518.

[17] Anindya Bijoy Das, Aditya Ramamoorthy, and Namrata Vaswani, "Efficient and robust distributed matrix computations via convolutional coding," *IEEE Trans. on Info. Th.*, vol. 67, no. 9, pp. 6266–6282, 2021.

[18] Seyyedali Hosseinalipour, Christopher G Brinton, Vaneet Aggarwal, Huaiyu Dai, and Mung Chiang, "From federated to fog learning: Distributed machine learning over heterogeneous wireless networks," *IEEE Comm. Mag.*, vol. 58, no. 12, pp. 41–47, 2020.

[19] Satyavrat Wagle, Seyyedali Hosseinalipour, Naji Khosravan, Mung Chiang, and Christopher G Brinton, "Embedding alignment for unsupervised federated learning via smart data exchange," in *Proc. of IEEE Glob. Comm. Conf.* IEEE, 2022, pp. 1–6.

[20] Naoya Yoshida, Takayuki Nishio, Masahiro Morikura, Koji Yamamoto, and Ryo Yonetani, "Hybrid-fl for wireless networks: Cooperative learning mechanism using non-iid data," in *Proc. of IEEE Intl. Conf. Comm.* IEEE, 2020, pp. 1–7.

[21] JR Marshall. Hall, *Combinatorial theory*, Wiley, 1986.

[22] Jacob T Schwartz, "Fast probabilistic algorithms for verification of polynomial identities," *Jour. of the ACM (JACM)*, vol. 27, no. 4, pp. 701–717, 1980.