Coded Computing for Low-Latency Federated Learning Over Wireless Edge Networks

Saurav Prakash, *Graduate Student Member, IEEE*, Sagar Dhakal, *Senior Member, IEEE*, Mustafa Riza Akdeniz, *Member, IEEE*, Yair Yona, Shilpa Talwar, *Member, IEEE*, Salman Avestimehr, *Fellow, IEEE*, and Nageen Himayat, *Member, IEEE*

***Abstract*— Federated learning enables training a global model from data located at the client nodes, without data sharing and moving client data to a centralized server. Performance of federated learning in a multi-access edge computing (MEC) network suffers from slow convergence due to heterogeneity and stochastic fluctuations in compute power and communication link qualities across clients. We propose a novel *coded computing* framework, CodedFedL, that injects structured coding redundancy into federated learning for mitigating stragglers and speeding up the training procedure. CodedFedL enables coded computing for non-linear federated learning by efficiently exploiting *distributed kernel embedding* via random Fourier features that transforms the training task into computationally favourable distributed linear regression. Furthermore, clients generate *local parity* datasets by coding over their local datasets, while the server combines them to obtain the *global parity* dataset. Gradient from the global parity dataset compensates for straggling gradients during training, and thereby speeds up convergence. For minimizing the *epoch deadline time* at the MEC server, we provide a tractable approach for finding the amount of coding redundancy and the number of local data points that a client processes during training, by exploiting the statistical properties of compute as well as communication delays. We also characterize the leakage in data privacy when clients share their local parity datasets with the server. Additionally, we analyze the convergence rate and iteration complexity of CodedFedL under simplifying assumptions, by treating CodedFedL as a stochastic gradient descent algorithm. Finally, for demonstrating gains that CodedFedL can achieve in practice, we conduct numerical experiments using practical network parameters and benchmark datasets, in which CodedFedL speeds up the overall training time by up to $15\times$ in comparison to the benchmark schemes.**

Manuscript received July 17, 2020; revised September 28, 2020; accepted October 26, 2020. Date of publication November 9, 2020; date of current version December 16, 2020. A part of this article is adapted from the paper presented at the Wireless Edge Intelligence Workshop, IEEE Globecom 2019 [1]. A part of this article was presented at the International Workshop on Federated Learning for User Privacy and Data Confidentiality, in Conjunction with ICML 2020 (FL-ICML'20) [2]. *(Corresponding author: Saurav Prakash.)*

Saurav Prakash and Salman Avestimehr are with Electrical Engineering Department, University of Southern California, Los Angeles, CA 90089 USA (e-mail: sauravpr@usc.edu; avestimehr@ee.usc.edu).

Sagar Dhakal was with Intel Corporation, Santa Clara, CA 95054-1549 USA. He is now with Broadcom Corporation, San Jose, CA 95131 USA (e-mail: sagar.dhakal@broadcom.com).

Mustafa Riza Akdeniz, Shilpa Talwar, and Nageen Himayat are with Intel Laboratories, Santa Clara, CA 95054 USA (e-mail: mustafa.akdeniz@intel.com; shilpa.talwar@intel.com; nageen.himayat@intel.com).

Yair Yona was with Intel Corporation, Santa Clara, CA 95054-1549 USA. He is now with Qualcomm, San Jose, CA 95110 USA (e-mail: yyona@qti.qualcomm.com).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/JSAC.2020.3036961.

Digital Object Identifier 10.1109/JSAC.2020.3036961

***Index Terms*— Distributed computing, machine learning, edge computing, wireless communication.**

I. Introduction

MASSIVE amounts of data are generated each day by the Internet of Things comprising billions of devices including autonomous vehicles, cell phones, and personal wearables [3]. This big data has the potential to power a wide range of statistical machine learning based applications such as predicting health events like a heart attack from wearable devices [4]. To enable low-latency and efficient computing capabilities close to the user traffic, there have been significant efforts recently to develop multi-access edge computing (MEC) platforms [5]–[9].

In classical MEC settings, client data is transferred to an underlying centralized computational infrastructure for further processing. However, client data can be of a personalized nature due to which there is an increasing privacy concern in moving the client data to a central location for any model training. For example, a person may want to use a machine learning application to predict health events like low sugar, but may not be willing to share the health records.

Federated learning framework has been recently developed to carry out machine learning tasks from data distributed at the client nodes, while the raw data is kept at the clients and never uploaded to the central server [10], [11]. As first formulated in [10], federated learning proceeds in two major steps. First, every client carries out a local gradient update on its local dataset. Second, a central server collects and aggregates the updates from the clients, updates the global model, and transmits it to the clients. The iterative procedure is carried out until convergence.

Implementation of federated learning in MEC networks suffers from some fundamental bottlenecks. The heterogeneity of compute and communication resources across clients makes the client selection a difficult task as the overall gradient aggregation at the MEC server can be significantly delayed by the straggling computations and communication links (see Figure 1). Additionally, federated learning suffers from wireless link failures during transmission. Re-transmission of messages can be done for failed communications, but it may drastically prolong the training time. Furthermore, the data distribution across clients in MEC networks is non-IID (non-independent and identically distributed), i.e. data stored

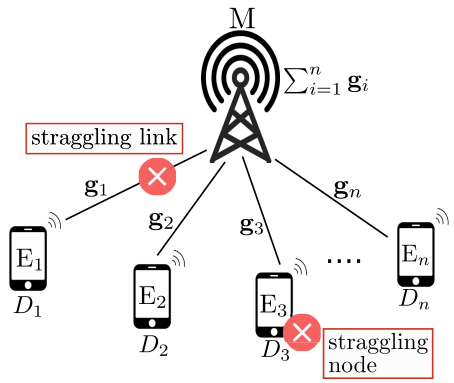


Fig. 1. Illustration of the federated learning paradigm over multi-access edge computing (MEC) networks with n client devices and an MEC server. During each training round, client E_j receives the latest model from server M, computes a local gradient update over its local dataset D_j, and communicates the gradient update to the server. Training performance is critically bottlenecked by the presence of straggling nodes and communication links.

locally on a device does not represent the population distribution [12]. Thus, missing out updates from clients leads to poor convergence.

CodedFedL Overview: To overcome the aforementioned challenges, we propose *CodedFedL*, a novel *coded computing* framework that leverages coding theoretic ideas to inject structured redundancy in federated learning for mitigating straggling clients and communication links, and improving performance of federated learning with non-IID data. In the following, we summarize the key aspects of our proposal.

- *Coded Computation at the MEC Server*: CodedFedL leverages the compute power of the federated learning server. Particularly, for distributed linear regression, we propose to generate masked parity data locally at each client at the start of the training procedure, by taking linear combinations of features and labels in the local dataset. The encoding coefficients are locally generated by the clients, and along with the raw client data, the encoding coefficients are not shared with the server. The local parity datasets are shared with the server, which aggregates them to obtain the composite global parity dataset. During training, the central server obtains the coded gradient by computing the gradient over the global parity data, which compensates for the erased or delayed parameter updates from the straggling clients. The combination of coded gradient computed by the MEC server and the uncoded gradients from the non-straggling clients stochastically approximates the full gradient over the entire dataset available at the clients, thus mitigating the convergence issues arising due to missing out updates from clients when data is non-IID.
- *Non-linear Federated Learning*: For enabling non-linear model training, we propose to have a data pre-processing step that transforms the distributed learning task into linear regression, by leveraging the popular kernel embedding based on random Fourier features (RFF) [13]. Each client then generates its parity dataset by taking linear combinations over its transformed features and associated labels, and the server combines them to obtain the global parity dataset. Training is then carried out with the transformed dataset at the clients and the global parity dataset at the server, as outlined in the previous bullet.
- *Optimal Load Allocation*: For obtaining the amount of coding redundancy and the number of local data points that a client processes during training, we formulate an optimization problem to find the minimum deadline time until which the MEC server should wait in each round before updating the model. We provide an analytical and tractable approach for efficiently finding the coding redundancy and load allocation that optimizes the deadline time. Our approach is based on solving a key subproblem, which can be cast as a piece-wise convex optimization problem with bounded domain and hence can be solved efficiently using standard convex optimization tools. We also derive the unique closed form solution for this subproblem for a special case in which the communication links are fully reliable with adequate error protection coding, thus covering the special case of the AWGN (Additive White Gaussian Noise) channel.
- *Privacy Characterization*: For characterizing the privacy leakage in sharing local parity datasets with the server, we consider the case when each client utilizes an encoding matrix whose entries are independently drawn from a standard normal distribution. We consider the notion of ϵ-mutual-information differential privacy (MI-DP), that is closely related to differential privacy [14]. Specifically, we leverage the recent result in [15] for the ϵ-MI-DP of Gaussian random projections, and bound the leakage in a client's data privacy as a function of its database and the size of the parity dataset.
- *Convergence Analysis*: In CodedFedL, the expectation of the combination of the coded gradient and the uncoded gradients that the MEC server receives by the optimal deadline time is approximately equal to the full gradient over the entire dataset at the clients. Under simplifying assumptions, we analyze convergence and quantify the iteration complexity of CodedFedL, by treating the learning process as a stochastic gradient descent algorithm.
- *Performance Results*: We evaluate performance gains of CodedFedL by carrying out numerical experiments with a wireless MEC setting, benchmark datasets and non-IID data across clients. We consider the naive uncoded baseline where the server waits for all client updates, as well as the greedy uncoded baseline, where the server waits for a subset of the client updates. For achieving the same target test accuracy, CodedFedL achieves significant gains in the wall-clock training time of up to $5.8\times$ over naive uncoded, and up to $15\times$ over greedy uncoded. Furthermore, for identical number of training iterations, CodedFedL achieves almost the same test accuracy as the naive uncoded, while it outperforms greedy uncoded by an absolute accuracy margin of up to $\sim 13\%$, demonstrating the superiority of CodedFedL with non-IID data.

Related Works: A common system approach for straggler mitigation in distributed computing has been the introduction of some form of task replication [16], [17]. Recently, coded computing strategies have been developed for injecting

computation redundancy in unorthodox encoded forms to efficiently deal with communication bottleneck and system disturbances like stragglers, outages and node failures in distributed systems [18]–[25]. Particularly, [19] proposed to use erasure coding for speeding up distributed matrix multiplication and linear regression tasks. Coding for heterogeneous distributed matrix multiplication is proposed in [20], which developed an analytical method to calculate near-optimal coding redundancy. However, the entire data needs to be centrally encoded by the central server before assigning portions to compute devices. Reference [21] proposed a coding method over gradients for synchronous gradient descent, while [26] proposed to encode over the data for avoiding the impact of stragglers for linear regression tasks. Many other works on coded computing for straggler mitigation in distributed learning have been proposed in the recent past [27]–[31]. In all these works, the data placement and coding strategy is orchestrated by a central server. As a result, these works are not applicable in the federated learning setting, where the data is privately owned by the clients and cannot be shared with a central server. Our proposed coded computing framework, CodedFedL, provides a novel solution for leveraging coding redundancy for straggler resilient federated learning.

Prior works that have considered one or more aspects of compute, communication and statistical heterogeneity across clients in federated learning include [32]–[35]. In [32], a FedProx algorithm was proposed to address non-IID data across clients. However, [32] did not consider variability of compute and communication capabilities across clients. Reference [33] proposed FEDL algorithm for allocating radio resources to clients for reducing convergence time. However, we consider the MEC setting with personalized devices where the compute and communication resources of the clients cannot be tuned. In [34], important clients are selected based on compute and communication delays as well as importance of data in each round of training. In contrast, we optimize load allocation and coding redundancy only once at the start of training. Furthermore, these works do not leverage the computing capability of the MEC server. In [35], the authors propose a cooperative mechanism in which a fraction of clients share potentially all of their raw data with the server, which carries out gradient computations and includes them in model updates to mitigate statistical heterogeneity. However, sharing potentially all of the raw data from even a fraction of clients may not be feasible in the privacy sensitive federated learning paradigm. Additionally, the success of [35] depends on whether the clients that agree to share their raw data with the server adequately represent all the classes. This may not be practical as clients owning a certain type of data (such as users suffering from certain diseases) may not agree to participate in sharing of raw data. In CodedFedL, the server obtains a global parity dataset at the start of training via distributed encoding across client data, as each client privately encodes over its local dataset. In each training round, the server computes a coded gradient over the parity dataset that allows the central server to mitigate the impact of straggling nodes during training by stochastically approximating the gradient over the entire dataset across the clients.

TABLE I
MAIN NOTATIONS

n	number of client nodes
d	dimension of raw feature space
q	dimension of transformed feature space
$[n]$, $n\in\mathbb{N}$	set $\{1,\ldots,n\}$
ℓ_j, $j\in[n]$	number of data points in j-th client's dataset D_j
m	number of data points across all clients
$\boldsymbol{\theta}^r$, $r\in\{1,2,\ldots\}$	global model after r-th training iteration

We organize the rest of our paper as follows. In Table I, we list the main notations for convenience. Section II presents a technical background on federated learning, and our proposed compute and communication models for MEC. Section III describes our proposed CodedFedL scheme. In Section IV, we analyze the load allocation policy in CodedFedL. We provide the results of our numerical experiments in Section V, and provide our concluding remarks in Section VI. All technical proofs are provided in the Appendix.

II. Problem Setup and MEC Model

In this section, we first describe the federated learning setting, and consider linear regression as well as non-linear regression via kernel embedding. We then present our compute and communication models for MEC.

A. Federated Learning

There are n client nodes, each connected to the federated learning server. Client $j\in[n]$ has a local dataset $D_j = (\mathbf{X}^{(j)}, \mathbf{Y}^{(j)})$, where $\mathbf{X}^{(j)}\in\mathbb{R}^{\ell_j\times d}$ and $\mathbf{Y}^{(j)}\in\mathbb{R}^{\ell_j\times c}$ denote the feature set and the label set respectively as follows:

$$\begin{aligned}\mathbf{X}^{(j)} &= [\mathbf{x}_1^{(j)T},\ldots,\mathbf{x}_{\ell_j}^{(j)T}]^T,\\ \mathbf{Y}^{(j)} &= [\mathbf{y}_1^{(j)T},\ldots,\mathbf{y}_{\ell_j}^{(j)T}]^T.\end{aligned} \tag{1}$$

Here, $\ell_j = |D_j|$ denotes the number of feature-label tuples in D_j, while each data feature $\mathbf{x}_k^{(j)}\in\mathbb{R}^{1\times d}$, and its corresponding label $\mathbf{y}_k^{(j)}\in\mathbb{R}^{1\times c}$ for $k\in[\ell_j]$. Client $j\in[n]$ does not share its dataset D_j with the central server due to privacy concerns.

The goal in federated learning is to train a model by leveraging the data located at the clients. Specifically, the following general problem is considered:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}\in\mathcal{W}}{\arg\min}\,\frac{1}{m}\sum_{j=1}^{n}\sum_{k=1}^{\ell_j} l\left(\boldsymbol{\theta};(\mathbf{x}_k^{(j)},\mathbf{y}_k^{(j)})\right), \tag{2}$$

where $l(\boldsymbol{\theta};(\mathbf{x}_k^{(j)},\mathbf{y}_k^{(j)}))\in\mathbb{R}$ is the predictive loss associated with $(\mathbf{x}_k^{(j)},\mathbf{y}_k^{(j)})$ for model parameter $\boldsymbol{\theta}\in\mathcal{W}$, $m=\sum_{j=1}^{n}\ell_j$ denotes the total size of the dataset distributed across the clients, and $\mathcal{W}$ denotes the model parameter space. The solution to (2) is obtained via an iterative training procedure involving gradient descent. Specifically, in iteration $(r+1)$, server shares the current model $\boldsymbol{\theta}^{(r)}$ with the clients. Client $j\in[n]$ then computes the local gradient $\mathbf{g}^{(j)}$ as follows:

$$\mathbf{g}^{(j)} = \frac{1}{\ell_j}\sum_{k=1}^{\ell_j}\nabla_{\boldsymbol{\theta}} l\left(\boldsymbol{\theta}^{(r)};(\mathbf{x}_k^{(j)},\mathbf{y}_k^{(j)})\right). \tag{3}$$

The server collects gradients from the clients and aggregates them to recover the gradient of the empirical loss corresponding to the entire distributed dataset across clients as follows:

$$\mathbf{g} = \frac{1}{m}\sum_{j=1}^{n} \ell_j \mathbf{g}^{(j)}. \tag{4}$$

The server then executes the model update step as follows:

$$\boldsymbol{\theta}^{(r+1)} = \boldsymbol{\theta}^{(r)} - \mu^{(r+1)}\mathbf{g} \tag{5}$$

where $\mu^{(r+1)}$ denotes the learning rate. The iterative procedure is carried out until sufficient convergence has been achieved.

For linear regression with squared error loss, the optimization problem in (2) is cast as follows:

$$\begin{aligned}\boldsymbol{\theta}^* &= \underset{\boldsymbol{\theta}\in\mathbb{R}^{d\times c}}{\arg\min} \frac{1}{2m}\sum_{j=1}^{n}\sum_{k=1}^{\ell_j} \|\mathbf{x}_k^{(j)}\boldsymbol{\theta} - \mathbf{y}_k^{(j)}\|_2^2, \\ &= \underset{\boldsymbol{\theta}\in\mathbb{R}^{d\times c}}{\arg\min} \frac{1}{m}\sum_{j=1}^{n} \frac{1}{2}\|\mathbf{X}^{(j)}\boldsymbol{\theta} - \mathbf{Y}^{(j)}\|_F^2, \end{aligned} \tag{6}$$

and the local gradient computation at client $j\in[n]$ is as follows:

$$\begin{aligned}\mathbf{g}^{(j)} &= \frac{1}{2\ell_j}\nabla_{\boldsymbol{\theta}}\|\mathbf{X}^{(j)}\boldsymbol{\theta}^{(r)} - \mathbf{Y}^{(j)}\|_F^2 \\ &= \frac{1}{\ell_j}\mathbf{X}^{(j)T}(\mathbf{X}^{(j)}\boldsymbol{\theta}^{(r)} - \mathbf{Y}^{(j)}). \end{aligned} \tag{7}$$

The gradient aggregation and model update steps in (4) and (5) are modified accordingly.

Linear regression has been traditionally used widely in a variety of applications including weather data analysis, market research studies and observational astronomy [36]–[38]. As evident from (7), the gradient computations involve matrix multiplications which are computationally favourable, particularly for low powered client devices with limited compute capabilities. However, in many machine learning problems, a linear model does not perform well.

To combine the advantages of non-linear models and low complexity gradient computations in linear regression, random Fourier feature mapping (RFFM) [13] based kernel regression has been widely used in practice. RFFM proposed in [13] involves explicitly constructing finite-dimensional random features from the raw features, such that the inner product of any pair of transformed features approximates the kernel evaluation corresponding to the two raw features. Specifically, features $\mathbf{v}_1\in\mathbb{R}^{1\times d}$ and $\mathbf{v}_2\in\mathbb{R}^{1\times d}$ are mapped to $\widehat{\mathbf{v}}_1$ and $\widehat{\mathbf{v}}_2$ using a feature generating function $\phi : \mathbb{R}^{1\times d}\rightarrow\mathbb{R}^{1\times q}$. The RFF mapping approximates a positive definitive kernel function $K : \mathbb{R}^{1\times d}\times\mathbb{R}^{1\times d}\rightarrow\mathbb{R}$ as represented below:

$$K(\mathbf{v}_1, \mathbf{v}_2) \approx \widehat{\mathbf{v}}_1\widehat{\mathbf{v}}_2^T = \phi(\mathbf{v}_1)\phi(\mathbf{v}_2)^T. \tag{8}$$

In Section III-A, we propose how to carry out distributed kernel embedding, so that client $j\in[n]$ transforms its dataset $D_j = (\mathbf{X}^{(j)}, \mathbf{Y}^{(j)})$ to $\widehat{D}_j = (\widehat{\mathbf{X}}^{(j)}, \mathbf{Y}^{(j)})$, where $\widehat{\mathbf{X}}^{(j)} = [\phi(\mathbf{x}_1^{(j)})^T, \ldots, \phi(\mathbf{x}_{\ell_j}^{(j)})^T]^T$ denotes the transformed feature set. Training is then performed via linear regression over the transformed data located at the clients, i.e., the optimization problem in (2) is cast as follows:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}\in\mathbb{R}^{q\times c}}{\arg\min} \frac{1}{2m}\sum_{j=1}^{n}\|\widehat{\mathbf{X}}^{(j)}\boldsymbol{\theta} - \mathbf{Y}^{(j)}\|_F^2, \tag{9}$$

while the gradient computation in iteration $(r+1)$ at client $j\in[n]$ is as follows:

$$\mathbf{g}^{(j)} = \frac{1}{\ell_j}\widehat{\mathbf{X}}^{(j)T}(\widehat{\mathbf{X}}^{(j)}\boldsymbol{\theta}^{(r)} - \mathbf{Y}^{(j)}). \tag{10}$$

Gradient aggregation and model update steps are then carried out at the server according to (4) and (5) respectively.

Remark 1: Training and inference with random features have been shown to work considerably well in practice [13], [39]–[42]. Hence, without loss of generality, we focus on non-linear federated learning via kernel regression with RFF mapping in the remaining part of our paper. All results easily generalize to plain linear regression.

To capture the heterogeneity and stochastic nature of compute and communication capabilities across clients in MEC networks, we consider probabilistic models as described next.

B. Compute and Communication Models

To statistically represent compute heterogeneity, we assume a shifted exponential model for local gradient computation. Specifically, the computation time for j-th client is given by a shifted exponential random variable $T_{cmp}^{(j)}$ as follows:

$$T_{cmp}^{(j)} = T_{cmp}^{(j,1)} + T_{cmp}^{(j,2)}. \tag{11}$$

Here, $T_{cmp}^{(j,1)} = \frac{\widetilde{\ell}_j}{\mu_j}$ denotes the time in seconds to process the partial gradient over $\widetilde{\ell}_j$ data points, where data processing rate is μ_j data points per second, and $\widetilde{\ell}_j$ is bounded by the size of the local dataset $\widehat{D}_j$, i.e., $\widetilde{\ell}_j\leq\ell_j$. $T_{cmp}^{(j,2)}$ is the random variable denoting the stochastic component of compute time coming from random memory access during read/write cycles associated with Multiply-Accumulate (MAC) operations, where we assume an exponential distribution for $T_{cmp}^{(j,2)}$, i.e., $p_{T_{cmp}^{(j,2)}}(t) = \gamma_j e^{-\gamma_j t}$, $t\geq 0$, where $\gamma_j = \frac{\alpha_j\mu_j}{\widetilde{\ell}_j}$. The parameter $\alpha_j > 0$ controls the ratio of the time spent in computing to the average time spent in memory access.

In addition to the local computation time $T_{cmp}^{(j)}$, the overall execution time for j-th client during $(r+1)$-th epoch includes $T_{com-d}^{(j)}$, time to download $\boldsymbol{\theta}^{(r)}$ from the server, and $T_{com-u}^{(j)}$, time to upload the partial gradient $\mathbf{g}^{(j)}$ to the server. We assume that communications between server and clients take place over wireless links that fluctuate in quality. It is a typical practice to model the wireless link between server and j-th client by a tuple (η_j, p_j), where η_j and p_j denote the achievable data rate (in bits per second per Hz) and link erasure probability p_j [43]–[45]. Downlink and uplink communication delays are IID random variables given as follows:[1]

$$T_{com-d}^{(j)} = N_j^d\tau_j,\ T_{com-u}^{(j)} = N_j^u\tau_j \tag{12}$$

[1]For the purpose of this article, we assume the downlink and the uplink delays to be reciprocal. Generalization of our framework to asymmetric delay model is easy to address.

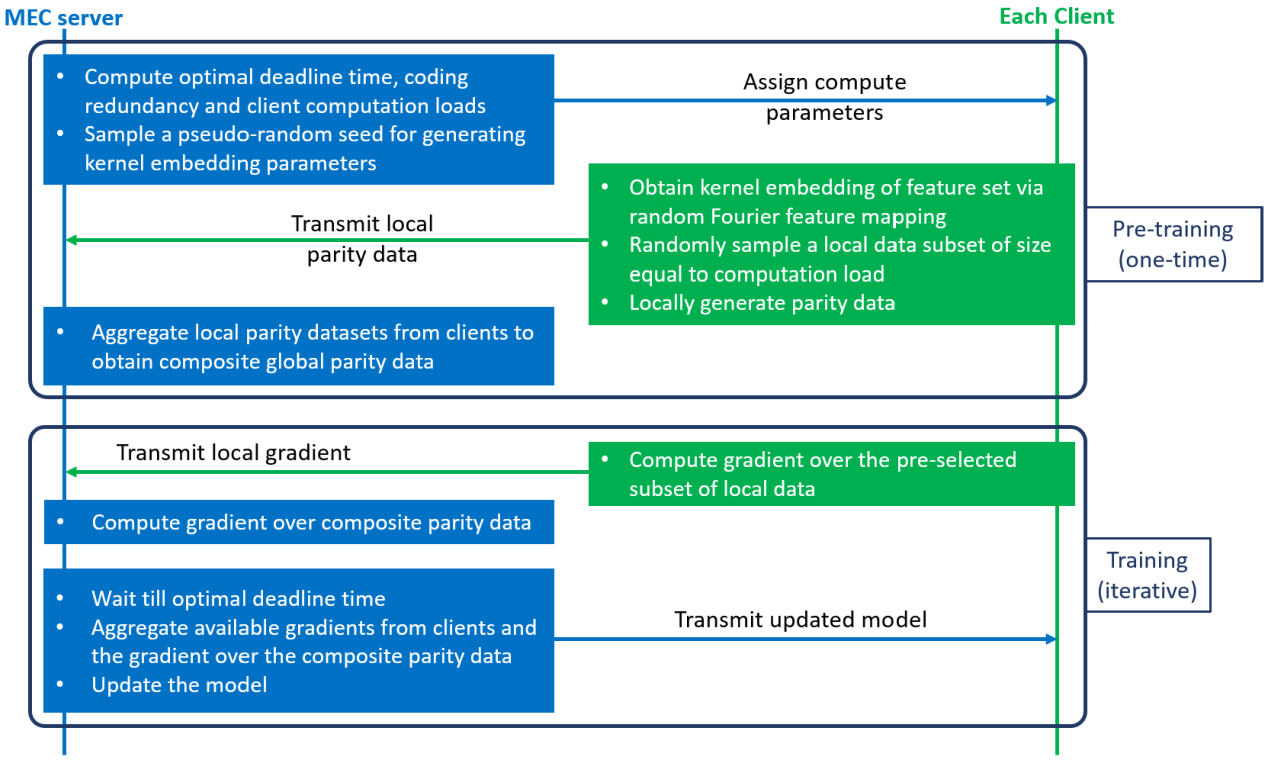


Fig. 2. Overview of our proposed CodedFedL framework, illustrating the main processing steps at the MEC server and at each client.

Here, $\tau_j = \frac{b}{\eta_j W}$ is the deterministic time to upload (or download) a packet of size b bits containing partial gradient $\mathbf{g}^{(j)}$ (or model $\boldsymbol{\theta}^{(r)}$) and W is the bandwidth in Hz assigned to the j-th worker device. N_j^d and N_j^u, that denote the number of transmissions required for successful downlink and uplink communications respectively, are distributed IID according to $Geometric(p = 1 - p_j)$ distribution as follows:

$$\begin{aligned}\mathbb{P}(N_j^d = x) &= \mathbb{P}(N_j^u = x)\\ &= p_j^{x-1}(1-p_j), \quad x = 1, 2, 3, \dots \end{aligned} \tag{13}$$

Therefore, using (11) and (12), the total time that the j-th device takes to successfully receive the latest model, complete its local gradient computation, and communicate the gradient to the central server, is as follows:

$$T_j = T_{com-d}^{(j)} + T_{cmp}^{(j)} + T_{com-u}^{(j)}, \tag{14}$$

while the average delay is given as follows:

$$\mathbb{E}(T_j) = \frac{\widetilde{\ell}_j}{\mu_j}\left(1 + \frac{1}{\alpha_j}\right) + \frac{2\tau_j}{1-p_j}. \tag{15}$$

The federated learning procedure can be severely impacted by slow nodes, straggling communication links, and non-IID data across clients. In the following section, we describe our proposed coded computing framework, CodedFedL, that injects structured redundancy into the federated learning procedure over MEC networks for mitigating these challenges.

III. Proposed CodedFedL Scheme

We now describe the different modules of our proposed CodedFedL scheme: distributed feature mapping for non-linear regression, distributed encoding for generating composite parity data, optimal load allocation and code design for minimizing deadline time, and modified training at the MEC server. An overview of CodedFedL is provided in Fig. 2.

A. Distributed Kernel Embedding

For combining the advantages of superior performance of non-linear models and low computational complexity of gradient computations in linear regression, we propose to leverage kernel embedding based on random Fourier feature mapping (RFFM) in federated learning. Let $D = \cup_{j=1}^{n} D_j = (\mathbf{X}, \mathbf{Y})$ denote the entire dataset located at the clients, where $\mathbf{X} \in \mathbb{R}^{m \times d}$ and $\mathbf{Y} \in \mathbb{R}^{m \times c}$ respectively denote the combined feature and label sets at the clients as follows:

$$\mathbf{X} = [\mathbf{X}^{(1)T}, \dots, \mathbf{X}^{(n)T}]^T, \quad \mathbf{Y} = [\mathbf{Y}^{(1)T}, \dots, \mathbf{Y}^{(n)T}]^T. \tag{16}$$

In this paper, we consider the commonly used kernel known as the Radial Basis Function (RBF) kernel [46], in which for features $\mathbf{v}_1 \in \mathbf{X}$ and $\mathbf{v}_2 \in \mathbf{X}$, the following relationship holds:

$$K(\mathbf{v}_1, \mathbf{v}_2) = e^{-\frac{\|\mathbf{v}_1 - \mathbf{v}_2\|^2}{2\sigma^2}}, \tag{17}$$

where σ is a kernel hyperparameter. For $i \in [m]$, RFFM corresponding to the RBF kernel can be carried out for feature vector $\mathbf{v}_i$ as follows (see Section V, example (a) in [39]):

$$\begin{aligned}\widehat{\mathbf{v}}_i &= \phi(\mathbf{v}_i)\\ &= \sqrt{\frac{2}{q}}\left[\cos(\mathbf{v}_i\boldsymbol{\omega}_1 + \delta_1), \dots, \cos(\mathbf{v}_i\boldsymbol{\omega}_q + \delta_q)\right)] \end{aligned} \tag{18}$$

where for $s\in[q]$, the frequency vectors $\boldsymbol{\omega}_s\in\mathbb{R}^{d\times 1}$ are drawn independently from $\mathcal{N}(\mathbf{0}, \frac{1}{2\sigma^2}\mathbf{I}_d)$, while the shift elements δ_s are drawn independently from the $Uniform(0, 2\pi]$ distribution. Before commencing the training procedure, j-th client carries out RFFM on its raw feature set $\mathbf{X}^{(j)}$ to obtain the transformed feature set $\widehat{\mathbf{X}}^{(j)} = \phi(\mathbf{X}^{(j)})$, and the training proceeds with the transformed dataset $\widehat{D} = (\widehat{\mathbf{X}}, \mathbf{Y})$, where $\widehat{\mathbf{X}}\in\mathbb{R}^{m\times q}$ is the matrix denoting all the transformed features across all clients.

Remark 2: For distributed transformation of features at the clients, the server sends the same pseudo-random seed to every client which then obtains the samples required for RFFM in (18). This mitigates the need for the server to communicate the frequency vectors $\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_q$ and the shift elements $\delta_1, \ldots, \delta_q$ to each client, thus reducing the communication overhead of distributed kernel embedding significantly.

Along with the computational benefits of linear regression over the transformed dataset $\widehat{D} = (\widehat{\mathbf{X}}, \mathbf{Y})$, applying RFFM enables our distributed encoding strategy for creating global parity data for non-linear federated learning, as described next.

B. Distributed Encoding

To inject coding redundancy into federated learning, j-th client carries out random linear encoding of its transformed training dataset $\widehat{D}_j = (\widehat{\mathbf{X}}^{(j)}, \mathbf{Y}^{(j)})$. Specifically, random generator matrix $\mathbf{G}_j\in\mathbb{R}^{u\times \ell_j}$ is used for encoding, where the row dimension u denotes the *coding redundancy* which is the amount of parity data to be generated at each device. Typically, $u\ll m$. Our strategy to find the amount u of coding redundancy and the local computation loads of the clients is presented in Section III-C, where we describe our load allocation policy for optimizing the deadline time at the server.

Client $j\in[n]$ privately draws the elements of $\mathbf{G}_j$ independently from a probability distribution with mean 0 and variance 1. For example, it can use a standard normal $\mathcal{N}(0, 1)$ distribution, or a $Bernoulli(1/2)$ distribution with sample space $\{-1, +1\}$. Client j keeps the encoding matrix $\mathbf{G}_j$ private and does not share it with the server. $\mathbf{G}_j$ is applied on the $weighted$ local dataset to obtain the local parity dataset $\breve{D}_j = (\breve{\mathbf{X}}^{(j)}, \breve{\mathbf{Y}}^{(j)})$ as follows:

$$\breve{\mathbf{X}}^{(j)} = \mathbf{G}_j\mathbf{W}_j\widehat{\mathbf{X}}^{(j)}, \quad \breve{\mathbf{Y}}^{(j)} = \mathbf{G}_j\mathbf{W}_j\mathbf{Y}^{(j)}. \tag{19}$$

For $\mathbf{w}_j = [w_{j,1}, \ldots, w_{j,\ell_j}]$, the weight matrix $\mathbf{W}_j = \text{diag}(\mathbf{w}_j)$ is an $\ell_j\times\ell_j$ diagonal matrix that weighs the training data point $(\widehat{\mathbf{x}}_k^{(j)}, \mathbf{y}_k^{(j)})\in\widehat{D}_j$ with $w_{j,k}$, based on the stochastic conditions of the compute and communication resources, where $k\in[\ell_j]$. We defer the details of deriving $\mathbf{W}_j$ to Section III-D.

The central server receives the local parity data from all client devices and combines them to obtain the composite global parity dataset $\breve{D} = (\breve{\mathbf{X}}, \breve{\mathbf{Y}})$, where $\breve{\mathbf{X}}\in\mathbb{R}^{u\times q}$ and $\breve{\mathbf{Y}}\in\mathbb{R}^{u\times c}$ are the composite global parity feature set and global parity label set as follows:

$$\breve{\mathbf{X}} = \sum_{j=1}^{n}\breve{\mathbf{X}}^{(j)}, \quad \breve{\mathbf{Y}} = \sum_{j=1}^{n}\breve{\mathbf{Y}}^{(j)} \tag{20}$$

Using (19) and (20) we have the following:

$$\breve{\mathbf{X}} = \mathbf{G}\mathbf{W}\widehat{\mathbf{X}}, \quad \breve{\mathbf{Y}} = \mathbf{G}\mathbf{W}\mathbf{Y}, \tag{21}$$

where $\mathbf{G} = [\mathbf{G}_1, \ldots, \mathbf{G}_n]\in\mathbb{R}^{u\times m}$ is the global encoding matrix and $\mathbf{W}\in\mathbb{R}^{m\times m}$ is the global weight matrix given by $\mathbf{W} = \text{diag}([\mathbf{w}_1, \ldots, \mathbf{w}_n])$. Equation (21) thus represents encoding over *the entire decentralized dataset* $\widehat{D} = (\widehat{\mathbf{X}}, \mathbf{Y})$, performed implicitly in a distributed manner across clients.

Remark 3: Although client $j\in[n]$ shares its locally coded dataset $\breve{D}_j = (\breve{\mathbf{X}}^{(j)}, \breve{\mathbf{Y}}^{(j)})$ with the central server, the local dataset $\widehat{D}_j$ as well as the encoding matrix $\mathbf{G}_j$ are private to the client and not shared with the server. In Appendix F, we characterize the privacy leakage in sharing local parity dataset with the server.

Next, we describe our load allocation policy to minimize the epoch deadline time for receiving the gradient updates from the non-straggling nodes.

C. Coding Redundancy and Load Assignment

CodedFedL involves load optimization based on the statistical conditions of MEC for obtaining the minimum deadline time t^{opt}, and correspondingly an optimal number of data points $\ell_j^{opt}\leq\ell_j$ to be processed locally at client $j\in[n]$ in each round, as well as $u^{opt}\leq u^{max}$, the number of coded data points to be processed at the server in each round. Here, we assume that due to memory and storage constraints, the server can process a maximum of u^{max} coded data points in each round. Furthermore, for generality, we assume that the server offloads the computation to a high performance computing unit. During training, the computing unit receives the current model from the server, carries out gradient computations over its assigned coded data, and uploads the gradient to the server, where the communications to and from the server take place over a wireless channel. Therefore, to parameterize the compute and communication capabilities of the MEC server, we use similar compute and communication models as described in Section II-B. We let T_C denote the random variable for the overall time spent by the computing unit in receiving the current model from the server, computing the gradient over its assigned coded data, and communicating the gradient to the server. In practice, the MEC server has dedicated, high performance and reliable cloud like compute and communication resources [47], [48]. Thus, in comparison to client devices in practice, MEC server has higher values for the data processing rate μ and parameter α in the computation model in (11), a higher value of data transmission rate η and a lower value of channel failure probability p in the communication model in (13).

Let $\mathbb{1}_{\{T_j\leq t\}}$ be the indicator random variable denoting the event that the server receives the partial gradient over the $\widetilde{\ell}_j\leq\ell_j$ local data points from j-th client by the deadline time t, where T_j denotes the total delay for client $j\in[n]$. To represent this contribution from j-th client by deadline time t, we use the random variable $R_j(t;\widetilde{\ell}_j) = \widetilde{\ell}_j\mathbb{1}_{\{T_j\leq t\}}$. Clearly, $R_j(t;\widetilde{\ell}_j)\in\{0, \widetilde{\ell}_j\}$. We let $R_U(t;\widetilde{\boldsymbol{\ell}}) = \sum_{j=1}^{n}R_j(t;\widetilde{\ell}_j)$ denote the *uncoded* aggregate return from the clients till deadline time t. Similarly, for representing the completion of the gradient computation over the parity dataset $\breve{D} = (\breve{\mathbf{X}}, \breve{\mathbf{Y}})$ within deadline time t, we use the random variable $R_C(t;u) = u\mathbb{1}_{\{T_C\leq t\}}$, with $\mathbb{1}_{\{T_C\leq t\}}$ being the indicator random variable

denoting the event that the *coded gradient* is available for aggregation within deadline time t. Clearly, $R_C(t;u)\in\{0,u\}$. Then, the following denotes the *total aggregate return* for $t\geq 0$:

$$R(t;(u,\widetilde{\boldsymbol{\ell}})) = R_C(t;u) + R_U(t;\widetilde{\boldsymbol{\ell}}). \tag{22}$$

Our goal is to optimize over t, $\widetilde{\boldsymbol{\ell}} = (\widetilde{\ell}_1,\ldots,\widetilde{\ell}_n)$ and u such that the optimal expected total aggregate return is m for the minimum epoch deadline time, with m being the total number of data points at the clients. More formally, we consider the following optimization problem:

$$\begin{aligned} &\text{minimize } t \\ &\text{subject to } \mathbb{E}(R(t;(u,\widetilde{\boldsymbol{\ell}}))) = m, \\ &\qquad \mathbf{0} \leq \widetilde{\boldsymbol{\ell}} \leq (\ell_1,\ldots,\ell_n), \\ &\qquad 0 \leq u \leq u^{max}, \\ &\qquad t \geq 0. \end{aligned} \tag{23}$$

Let $(t^{opt}, u^{opt}, \boldsymbol{\ell}^{opt})$ denote an optimal solution for (23). In the following, we propose an efficient and tractable two-step approach for solving (23).

Step 1: First step is to optimize $\widetilde{\boldsymbol{\ell}}$ and u in order to maximize the expected return $\mathbb{E}(R(t;(u,\widetilde{\boldsymbol{\ell}})))$ for a fixed t. More precisely, for a given deadline time t, the goal is to solve the following for the total expected aggregate return:

$$\begin{aligned} &\text{maximize } \mathbb{E}(R(t;(u,\widetilde{\boldsymbol{\ell}}))) \\ &\text{subject to } \mathbf{0} \leq \widetilde{\boldsymbol{\ell}} \leq (\ell_1,\ldots,\ell_n) \\ &\qquad 0 \leq u \leq u^{max} \end{aligned} \tag{24}$$

As $\mathbb{E}(R(t;(u,\widetilde{\boldsymbol{\ell}}))) = \sum_{j=1}^{n}\mathbb{E}(R_j(t;\widetilde{\ell}_j)) + \mathbb{E}(R_C(t;u))$, the optimization in (24) can be decomposed into $(n+1)$ independent optimization problems, one for each client $j\in[n]$ as follows:

$$\begin{aligned} &\text{maximize } \mathbb{E}(R_j(t;\widetilde{\ell}_j)) \\ &\text{subject to } 0 \leq \widetilde{\ell}_j \leq \ell_j, \end{aligned} \tag{25}$$

and one for the MEC server as follows:

$$\begin{aligned} &\text{maximize } \mathbb{E}(R_C(t;u)) \\ &\text{subject to } 0 \leq u \leq u^{max}, \end{aligned} \tag{26}$$

Remark 4: In Section IV, we derive the mathematical expression for the expected return $\mathbb{E}(R_j(t;\widetilde{\ell}_j))$ for $j\in[n]$, and prove that it is a piece-wise concave function in $\widetilde{\ell}_j>0$. We also characterize the intervals within which the function is concave, and show that the boundaries are functions of the total number of transmissions needed for the successful downlink (model download) and uplink (gradient upload) communications by the deadline time t. Therefore, we can solve (25) efficiently using any convex optimization toolbox. The analysis follows similarly for (26). Therefore, (24) can be solved efficiently.

Let $\ell_j^*(t)$, for $j\in[n]$, and $u^*(t)$ denote optimal solutions of (25) and (26) respectively, which in turn optimize (24). Next, we describe the second step of our approach.

Step 2: Optimization of t is considered in order to find the minimum deadline time $t=t^*$ so that the maximized expected total aggregate return $\mathbb{E}(R(t;(u^*(t),\boldsymbol{\ell}^*(t))))$ is equal to m. Specifically, the following optimization problem is considered:

$$\begin{aligned} &\text{minimize } t \\ &\text{subject to } \mathbb{E}(R(t;(u^*(t),\boldsymbol{\ell}^*(t)))) = m, \\ &\qquad t \geq 0. \end{aligned} \tag{27}$$

Remark 5: We show that $\mathbb{E}(R(t;(u^*(t),\boldsymbol{\ell}^*(t))))$ is a monotonically increasing function in t in Section IV. Therefore, (27) can be efficiently solved to obtain t^* using a bisection search over t with a sufficiently large starting upper bound. Consequently, an optimal load allocation solution $(u^*(t^*),\boldsymbol{\ell}^*(t^*))$ is obtained as a solution of (24) for $t=t^*$.

Our proposed two-step load allocation strategy achieves an optimal solution of (23), as summarized in the following claim.

Claim: Let $(t^*,u^*(t^*),\boldsymbol{\ell}^*(t^*))$ be an optimal solution obtained by solving (27). Then, $t^* = t^{opt}$ and $(t^*,u^*(t^*),\boldsymbol{\ell}^*(t^*))$ is an optimal solution of (23).

Proof of the above claim is provided in Appendix A. In the next subsection, we describe the procedure used by client $j\in[n]$ for obtaining the weight matrix $\mathbf{W}_j$, which is used for generating the local parity dataset in (19).

D. Weight Matrix Construction

After the evaluation of the optimal load allocation $\boldsymbol{\ell}^*(t^*)$ for the clients described in the previous subsection, j-th client samples $\ell_j^*(t^*)$ data points uniformly and randomly that it will process for local gradient computation in each training round. It is not revealed to the server which data points are sampled. The probability that the partial gradient computed at j-th client is not received at the MEC server by deadline time t^* is $pnr_{j,1} = (1-\mathbb{P}(T_j\leq t^*))$. Furthermore, $(\ell_j - \ell_j^*(t^*))$ data points are never evaluated at the client, which implies that the probability of no return for them is $pnr_{j,2}=1$.

The diagonal weight matrix $\mathbf{W}_j\in\mathbb{R}^{\ell_j\times\ell_j}$, which is used for generating the local parity dataset in (19), captures the absence of the updates corresponding to the different data points during the training procedure. Specifically, for $k\in[\ell_j]$, if data point $(\widehat{\mathbf{x}}_k^{(j)},\mathbf{y}_k^{(j)})$ is among the $\ell_j^*(t^*)$ data points that are to be processed at the client during gradient computation, the corresponding weight matrix coefficient is $w_{j,k} = \sqrt{pnr_{j,1}}$, otherwise $w_{j,k}=\sqrt{pnr_{j,2}}$. As we illustrate next, this weighing ensures that the combination of the coded gradient and the uncoded gradient updates from the non-straggling clients *stochastically approximates* the full gradient $\mathbf{g}$ in (4) over the entire dataset $\widehat{D} = (\widehat{\mathbf{X}},\mathbf{Y})$ distributed across the clients.

E. Coded Federated Aggregation

In epoch $(r+1)$, the MEC server sends the current model $\boldsymbol{\theta}^{(r)}$ to the clients as well as its own computing unit for gradient computations, and waits until the optimal deadline time t^* before updating the model. The computing unit of the MEC server computes the coded gradient, which is the gradient over the composite parity data $\breve{D} = (\breve{\mathbf{X}},\breve{\mathbf{Y}})$, and the MEC server weighs it with a factor of $1/(1-pnr_C)$, where $pnr_C = (1-\mathbb{P}(T_C\leq t^*))$ denotes the probability of no return

for the coded gradient. Effectively, the coded gradient used by the MEC server during gradient aggregation can be represented as follows:

$$\mathbf{g}_C = \mathbb{1}_{\{T_C \le t^*\}} \frac{1}{(1-pnr_C)} \left(\frac{1}{u^*(t^*)} \breve{\mathbf{X}}^T (\breve{\mathbf{X}} \boldsymbol{\theta}^{(r)} - \breve{\mathbf{Y}}) \right)$$
$$= \frac{\mathbb{1}_{\{T_C \le t^*\}}}{(1-pnr_C)} \widehat{\mathbf{X}}^T \mathbf{W}^T \left(\frac{\mathbf{G}^T \mathbf{G}}{u^*(t^*)} \right) \mathbf{W} (\widehat{\mathbf{X}} \boldsymbol{\theta}^{(r)} - \mathbf{Y}), \quad (28)$$

where $\mathbb{1}_{\{T_C \le t^*\}}$ is the indicator random variable that denotes whether the coded gradient is available for aggregation by the optimal deadline t^* or not. As we describe soon, weighing the coded gradient by a factor of $\frac{1}{(1-pnr_C)}$ accounts for the averaging effect caused by the random variable $\mathbb{1}_{\{T_C \le t^*\}}$, and results in a stochastic approximation of the true gradient.

Similarly, each client $j \in [n]$ computes its partial gradient, and the server computes a weighted combination of the uncoded gradients received from the clients by the deadline time t^* as follows:

$$\mathbf{g}_U = \sum_{j=1}^{n} \ell_j^*(t^*) \mathbf{g}_U^{(j)},$$

where $\mathbf{g}_U^{(j)}$ represents the effective gradient contribution from client j by deadline time t^* as follows:

$$\mathbf{g}_U^{(j)} = \mathbb{1}_{\{T_j \le t^*\}} \frac{1}{\ell_j^*(t^*)} \widetilde{\mathbf{X}}^{(j)T} (\widetilde{\mathbf{X}}^{(j)} \boldsymbol{\theta}^{(r)} - \widetilde{\mathbf{Y}}^{(j)}), \quad (29)$$

Here, $\widetilde{D}_j = (\widetilde{\mathbf{X}}^{(j)}, \widetilde{\mathbf{Y}}^{(j)})$ is composed of the $\ell_j^*(t^*)$ data points that j-th client samples for processing before training. As we show soon, no further factors similar to $\frac{1}{(1-pnr_C)}$ in (28) are needed to be applied to the uncoded gradients as they are already accounted for during creation of the local parity datasets. Thus, the MEC server waits for the uncoded gradients from the clients and the coded gradient from its computing unit until the optimized deadline time t^*, and then aggregates $\mathbf{g}_C$ and $\mathbf{g}_U$ to obtain the *coded federated gradient* as follows:

$$\mathbf{g}_M = \frac{1}{m} (\mathbf{g}_C + \mathbf{g}_U). \quad (30)$$

The coded federated gradient $\mathbf{g}_M$ in (30) stochastically approximates the full gradient $\mathbf{g}$ in (4) for a sufficiently large coding redundancy $u^*(t^*)$, specifically, $\mathbb{E}(\mathbf{g}_M) \approx \mathbf{g}$. To verify this, we first observe that the following holds for the coded gradient $\mathbf{g}_C$ in (28):

$$\mathbb{E}(\mathbf{g}_C) = \frac{\mathbb{E}(\mathbb{1}_{\{T_C \le t^*\}})}{(1-pnr_C)} \widehat{\mathbf{X}}^T \mathbf{W}^T \left(\frac{\mathbf{G}^T \mathbf{G}}{u^*(t^*)} \right) \mathbf{W} (\widehat{\mathbf{X}} \boldsymbol{\theta}^{(r)} - \mathbf{Y})$$
$$\overset{(a)}{\approx} \widehat{\mathbf{X}}^T \mathbf{W}^T \mathbf{W} (\widehat{\mathbf{X}} \boldsymbol{\theta}^{(r)} - \mathbf{Y})$$
$$= \sum_{j=1}^{n} \sum_{k=1}^{\ell_j} w_{j,k}^2 \widehat{\mathbf{x}}_k^{(j)T} (\widehat{\mathbf{x}}_k^{(j)} \boldsymbol{\theta}^{(r)} - \mathbf{y}_k^{(j)}). \quad (31)$$

In (a), by using the weak law of large numbers, we have approximated the quantity $(\frac{1}{u^*(t^*)} \mathbf{G}^T \mathbf{G})$ by an identity matrix. This is a reasonable approximation for a sufficiently large coding redundancy $u^*(t^*)$, since each diagonal entry in $\frac{\mathbf{G}^T \mathbf{G}}{u^*(t^*)}$ converges to 1 in probability, while each non-diagonal entry converges to 0 in probability. Furthermore, as we demonstrate via numerical experiments in Section V, the convergence curve as a function of iteration for CodedFedL significantly overlaps that of the naive uncoded scheme where the server waits to aggregate the results of all the clients.

The expected aggregate gradient $\mathbb{E}(\mathbf{g}_U)$ from the clients received by the server by the deadline time t^* is as follows:

$$\mathbb{E}(\mathbf{g}_U) = \sum_{j=1}^{n} \mathbb{P}(T_j \le t^*) \widetilde{\mathbf{X}}^{(j)T} (\widetilde{\mathbf{X}}^{(j)} \boldsymbol{\theta}^{(r)} - \widetilde{\mathbf{Y}}^{(j)})$$
$$\overset{(a)}{=} \sum_{j=1}^{n} \sum_{\substack{k \in [\ell_j] \\ (\widehat{\mathbf{x}}_k^{(j)}, \mathbf{y}_k^{(j)}) \in \widetilde{D}_j}} \mathbb{P}(T_j \le t^*) \widehat{\mathbf{x}}_k^{(j)T} (\widehat{\mathbf{x}}_k^{(j)} \boldsymbol{\theta}^{(r)} - \mathbf{y}_k^{(j)})$$
$$\overset{(b)}{=} \sum_{j=1}^{n} \sum_{k=1}^{\ell_j} (1 - w_{j,k}^2) \widehat{\mathbf{x}}_k^{(j)T} (\widehat{\mathbf{x}}_k^{(j)} \boldsymbol{\theta}^{(r)} - \mathbf{y}_k^{(j)}), \quad (32)$$

where in (a), the inner sum denotes the sum over data points in $\widetilde{D}_j = (\widetilde{\mathbf{X}}^{(j)}, \widetilde{\mathbf{Y}}^{(j)})$, while in (b), all the points in the local dataset are included, with $(1 - w_{j,k}^2) = 0$ for the points in the set $\widehat{D}_j \backslash \widetilde{D}_j$. In light of (31) and (32), it follows that $\mathbb{E}(\mathbf{g}_M) \approx \mathbf{g}$.

Remark 6: In Appendix E, we perform convergence analysis of CodedFedL and find its iteration complexity under the simplifying assumption that $(\frac{1}{u^*(t^*)} \mathbf{G}^T \mathbf{G}) = \mathbf{I}_m$ which based on the above analysis, implies $\mathbb{E}(\mathbf{g}_M) = \mathbf{g}$. We bound the variance of $\mathbf{g}_M$ and leverage a standard result in literature for convergence of stochastic gradient descent. The exact convergence analysis taking into account the underlying distribution of the encoding matrix will be addressed in future work.

In the following section, we analyze the load allocation policy of CodedFedL, demonstrating how our proposed two-step load allocation problem in (27) can be solved efficiently.

IV. Analyzing CodedFedL Load Design

In this section, we demonstrate how our load allocation policy provides an efficient and tractable approach to obtaining the minimum deadline time in (23). For ease of notation, we index the clients and the MEC server using $j \in [n+1]$ throughout this section, where $j \in [n]$ denotes the n clients and $j = n+1$ denotes the MEC server, and use the generic term node for the MEC server as well as the clients. Likewise, $\widetilde{\ell}_{n+1} = u$, $\ell_{n+1} = u^{max}$, $\ell_{n+1}^{opt} = u^{opt}$, $\ell_{n+1}^*(t^*) = u^*(t^*)$, $T_{n+1} = T_C$, and $R_{n+1}(t; \widetilde{\ell}_{n+1}) = R_C(t; u)$. In the following, we present our result for the expected return $\mathbb{E}(R_j(t; \widetilde{\ell}_j))$ for node $j \in [n+1]$ as defined in Section III-C.

Theorem: For the compute and communication models defined in (11) and (13), let $0 \le \widetilde{\ell}_j \le \ell_j$ be the number of data points processed by node $j \in [n+1]$ in each training epoch. For a deadline time of t at the server, the expectation of the return $R_j(t; \widetilde{\ell}_j) = \widetilde{\ell}_j \mathbb{1}_{\{T_j \le t\}}$ satisfies the following:

$$\mathbb{E}(R_j(t; \widetilde{\ell}_j)) = \begin{cases} \sum_{\nu=2}^{\nu_m} U\left(t - \frac{\widetilde{\ell}_j}{\mu_j} - \tau_j \nu \right) h_\nu f_\nu(t; \widetilde{\ell}_j) & \text{if } \nu_m \ge 2 \\ 0 & \text{otherwise} \end{cases}$$

where $U(\cdot)$ is the unit step function with $U(x) = 1$ for $x > 0$ and 0 otherwise,

$$f_\nu(t; \widetilde{\ell}_j) = wtilde\ell_j \left(1 - e^{-\frac{\alpha_j \mu_j}{\widetilde{\ell}_j} (t - \frac{\widetilde{\ell}_j}{\mu_j} - \tau_j \nu)} \right),$$

$$h_\nu = (\nu-1)(1-p_j)^2 p_j^{\nu-2},$$

and $\nu_m \in \mathbb{Z}$ *satisfies* $t-\tau_j\nu_m>0$, $t-\tau_j(\nu_m+1)\leq 0$.

Proof of Theorem is provided in Appendix B. Next, we analyze the behavior of $\mathbb{E}(R_j(t;\widetilde{\ell}_j))$ for $\nu_m \geq 2$. For a given $t>0$ and $\nu\in\{2,\dots,\nu_m\}$, consider $f_\nu(t;\widetilde{\ell}_j)$ for $\widetilde{\ell}_j>0$. Then, the following holds:

$$f''_\nu(t;\widetilde{\ell}_j) = -e^{-\frac{\alpha_j\mu_j}{\ell_j}(t-\frac{\widetilde{\ell}_j}{\mu_j}-\nu\tau_j)}\frac{\alpha_j^2\mu_j{}^2(t-\nu\tau_j)^2}{\widetilde{\ell}_j^3} < 0.$$

Thus, $f_\nu(t;\widetilde{\ell}_j)$ is strictly concave in the domain $\widetilde{\ell}_j>0$. Furthermore, $f_\nu(t;\widetilde{\ell}_j)\leq 0$ for $\widetilde{\ell}_j \geq \mu_j(t-\tau_j\nu)$ for $\nu\in\{2,\dots,\nu_m\}$. Therefore, as highlighted in Remark 4, the expected return $\mathbb{E}(R_j(t;\widetilde{\ell}_j))$ is piece-wise concave in $\widetilde{\ell}_j$, and the exact intervals of concavity are $(0,\mu_j(t-\nu_m\tau_j)),\dots,(\mu_j(t-3\tau),\mu_j(t-2\tau))$. Furthermore, $\widetilde{\ell}_j$ is upper bounded by ℓ_j. Thus, for a given $t>0$, each of (25) and (26) decomposes into a finite number of convex optimization problems, that are efficiently solved in practice [49]. The piece-wise concave relationship between $\mathbb{E}(R_j(t;\widetilde{\ell}_j))$ and t is also illustrated in Fig. 3(a).

Consider an optimal solution $\ell_j^*(t)$ for node $j\in[n+1]$, and the corresponding optimized expected return $\mathbb{E}(R_j(t;\ell_j^*(t)))$. Intuitively, as we increase the deadline time t, optimized load $\ell_j^*(t)$ should vary such that the server receives more optimal expected return from node j. In Appendix C, we formally prove that $\mathbb{E}(R_j(t;\ell_j^*(t)))$ is monotonically increasing in the deadline time t. We also illustrate this in Fig. 3(b). Furthermore, as the maximal expected total aggregate return $\mathbb{E}(R(t;(\ell_1^*(t),\dots,\ell_{n+1}^*(t)))) = \sum_{j=1}^{n+1}\mathbb{E}(R_j(t;\ell_j^*(t)))$, the maximal expected total aggregate return is also monotonically increasing in t. Therefore, (27) can be solved efficiently using bisection search, as claimed earlier in Remark 5.

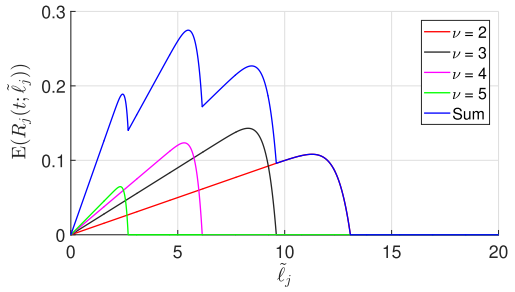
When communication links do not provide time diversity, as in AWGN channel, one reliable transmission is performed at less than 10^{-5} bit error rate with adequate error protection coding. This motivates us to consider the special case where for each node $j\in[n+1]$, $p_j=0$, resulting in the following specialized expression for the expected return:

$$\begin{aligned}\mathbb{E}(R_j(t;\widetilde{\ell}_j)) &= U\left(t-\frac{\widetilde{\ell}_j}{\mu_j}-2\tau_j\right)f_2(t;\widetilde{\ell}_j),\\ &= U\left(t-\frac{\widetilde{\ell}_j}{\mu_j}-2\tau_j\right)\widetilde{\ell}_j\left(1-e^{-\frac{\alpha_j\mu_j}{\ell_j}(t-\frac{\widetilde{\ell}_j}{\mu_j}-2\tau_j)}\right). \quad (33)\end{aligned}$$

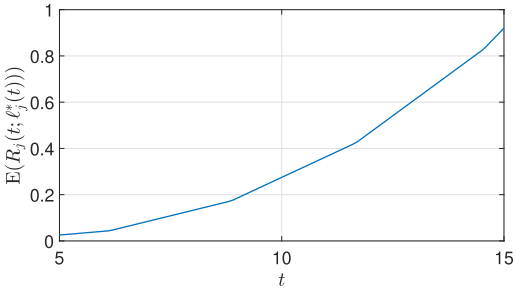
For this special case, we have a unique closed form solution for $\boldsymbol{\ell}^*(t) = (\ell_1^*(t),\dots,\ell_{n+1}^*(t))$, and consequently a closed form result for $\mathbb{E}(R(t;\boldsymbol{\ell}^*(t)))$. Specifically, for node $j\in[n+1]$, we have the following one-shot solution for the optimal load $\ell_j^*(t)$:

$$\ell_j^*(t) = \begin{cases}0 & \text{if } t\leq 2\tau_j\\ s_j(t-2\tau_j) & \text{if } 2\tau_j<t\leq\zeta_j\\ \ell_j & \text{otherwise}\end{cases} \quad (34)$$

where $s_j = -\frac{\alpha_j\mu_j}{W_{-1}\left(-e^{-(1+\alpha_j)}\right)+1}$ and $\zeta_j = (\frac{\ell_j}{s_j}+2\tau_j)$. Here, $W_{-1}(\cdot)$ is the minor branch of the Lambert W-function [50],



(a) Illustration of the piece-wise concavity of the expected return $\mathbb{E}(R_j(t;\widetilde{\ell}_j))$ for $\widetilde{\ell}_j>0$ for node j.



(b) Illustration of the monotonic relationship between optimal expected return $\mathbb{E}(R_j(t;\ell^*(t)))$ and t for j-th node.

Fig. 3. Illustrating the properties of expected aggregate return $\mathbb{E}(R_j(t;\widetilde{\ell}_j))$ based on the result in the Theorem. We assume $p_j=0.9$, $\tau_j=\sqrt{3}$, $\mu_j=2$, $\alpha_j=20$, and for Fig. 3(b), $t=10$.

which is the inverse function of $f(W)=We^W$. Consequently, we have the following one-shot solution for the optimized return for node $j\in[n+1]$:

$$\mathbb{E}(R_j(t;\ell_j^*(t))) = \begin{cases}0 & \text{if } t\leq 2\tau_j\\ \widetilde{s}_j(t-2\tau_j) & \text{if } 2\tau_j<t\leq\zeta_j\\ \ell_j\left(1-e^{-\frac{\alpha_j\mu_j}{\ell_j}\left(t-\frac{\ell_j}{\mu_j}-2\tau_j\right)}\right) & \text{otherwise}\end{cases} \quad (35)$$

where $\widetilde{s}_j = s_j(1-e^{-\alpha_j(\frac{\mu_j}{s_j}-1)})$. We prove (34) and (35) in Appendix D. Using these results, we have a closed form for the maximum expected total aggregate return from the nodes as follows:

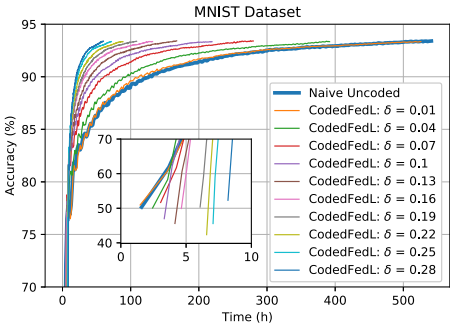
$$\mathbb{E}(R(t;\boldsymbol{\ell}^*(t))) = \sum_{\substack{j\in[n+1]\\ 2\tau_j<t\leq\zeta_j}}\widetilde{s}_j(t-2\tau_j) + \sum_{\substack{j\in[n+1]\\ \zeta_j<t}}\ell_j\left(1-e^{-\frac{\alpha_j\mu_j}{\ell_j}\left(t-\frac{\ell_j}{\mu_j}-2\tau_j\right)}\right), \quad (36)$$

which is monotonically increasing in the deadline time t. Therefore, (27) can be solved efficiently using bisection search to obtain the optimal deadline time t^*.

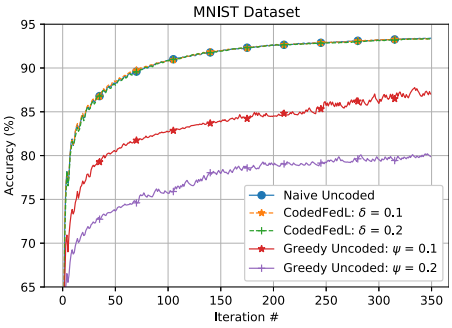
In the next section, we present the results of our numerical experiments, which demonstrate the performance gains that CodedFedL can achieve in practice.

V. Numerical Experiments

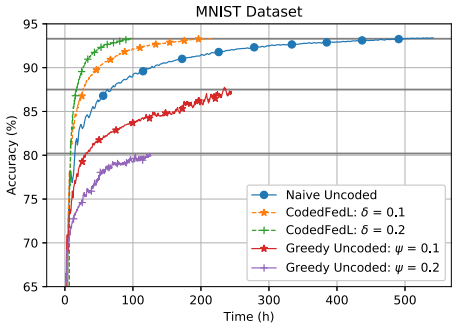
In this section, we demonstrate the performance gains of CodedFedL via numerical experiments. First we describe our simulation setting, and then we present the numerical results.



(a) Test accuracy with respect to wall-clock time for naive uncoded, and CodedFedL with different δ. Latency overhead due to uploading of the coded data is also highlighted.



(b) Test accuracy with respect to mini-batch update iteration for naive uncoded, greedy uncoded with $\psi \in \{0.1, 0.2\}$ and CodedFedL with $\delta \in \{0.1, 0.2\}$.



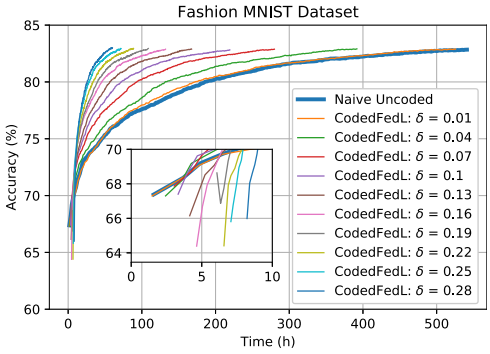
(c) Test accuracy with respect to the wall-clock time for naive uncoded, greedy uncoded with $\psi \in \{0.1, 0.2\}$ and CodedFedL with $\delta \in \{0.1, 0.2\}$.

Fig. 4. Illustrating the results for MNIST.

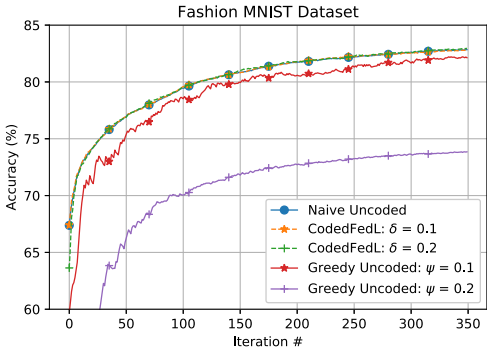
A. Simulation Setting

1) MEC Scenario: We consider a wireless scenario consisting of $n = 30$ client nodes and 1 MEC server. For each client, the delay model described in Section II-B is used for the overall time in downlink (downloading the model), gradient computation, and uplink (uploading the gradient update), where the system parameters are as described next. We use an LTE network, and assume that each client is uniformly allocated 3 resource blocks, resulting in a maximum PHY level information rate of 216 kbps. Note that depending on the channel conditions, the effective information rate can be lower than 216 kbps. To model heterogeneity, we generate normalized effective information rates using $\{1, k_1, k_1^2, \ldots, k_1^{29}\}$ and assign a random permutation of them to the clients, the maximum effective information rate being 216 kbps. Furthermore, we use the same failure probability $p_j = 0.1$ for $j \in \{1, \ldots, 30\}$, capturing the typical practice in wireless to adapt transmission rate for a constant failure probability [43]. An overhead of 10% is assumed and each scalar is represented by 32 bits. The normalized processing powers are generated using $\{1, k_2, k_2^2, \ldots, k_2^{29}\}$, the maximum MAC rate being 3.072×10^6 MAC/s. Furthermore, we set $\alpha_j = 2$ for $j \in \{1, \ldots, 30\}$. We fix $(k_1, k_2) = (0.95, 0.8)$. We assume that the MEC server has dedicated, high performance and reliable resources, so the coded gradient is available for aggregation with probability 1 by any finite deadline time t, i.e. $\mathbb{P}(T_C \leq t) = 1$ for any $t > 0$. Essentially, this implies $u^{opt} = u^{max}$. Furthermore, we let $\delta = u^{max}/m$ for notational convenience.

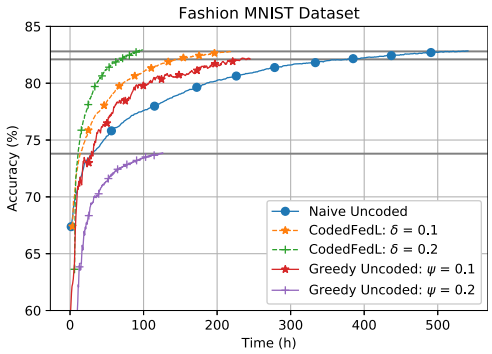
2) Datasets and Hyperparameters: We consider two benchmark datasets: MNIST [51] and Fashion MNIST [52]. The features are vectorized, and the labels are one-hot encoded. For kernel embedding, the hyperparameters are $(\sigma, q) = (5, 2000)$. A common practice in large-scale distributed learning is to perform training using mini-batch stochastic gradient descent (SGD), wherein the local dataset is first sorted and partitioned into mini-batches. Then, in each training iteration, each client computes gradient over a local mini-batch selected sequentially, and the model update is based on the gradient over the global mini-batch obtained by aggregating the gradients over the local mini-batches across clients. We consider the same mini-batch implementation for the uncoded schemes (as we describe in the next paragraph), while for CodedFedL, the data allocation, encoding and training modules are based on each global mini-batch. We assign equal number of data points to each client and use a global mini-batch size of $m = 12000$. Thus, each complete epoch over the training dataset constitutes 5 global mini-batch steps. For studying the impact of non-IID datasets across clients and demonstrate the superiority of CodedFedL in dealing with statistical heterogeneity, we first sort the training dataset according to class labels, and then partition the entire sorted training dataset into 30 equally sized shards, each of them to be assigned to a different worker. We then sort the clients according to the expected total time using the formula in (15) with $\widetilde{\ell}_j = 400$, i.e. the size of the local mini-batch. Then, the 30 data shards are allocated in the order of sorted clients. For all approaches, an initial step size of 6 is used with a step decay of 0.8 at epochs 40 and 65, while the total number of epochs is 70. Additionally, we use



(a) Test accuracy with respect to wall-clock time for naive uncoded, and CodedFedL with different δ. Latency overhead due to uploading of the coded data is also highlighted.



(b) Test accuracy with respect to mini-batch update iteration for naive uncoded, greedy uncoded with $\psi \in \{0.1, 0.2\}$ and CodedFedL with $\delta \in \{0.1, 0.2\}$.



(c) Test accuracy with respect to the wall-clock time for naive uncoded, greedy uncoded with $\psi \in \{0.1, 0.2\}$ and CodedFedL with $\delta \in \{0.1, 0.2\}$.

Fig. 5. Illustrating the results for Fashion MNIST.

an L_2 regularization of $\frac{\lambda}{2}\|\boldsymbol{\theta}\|_F^2$ with the loss defined in (9), and the regularization parameter λ is set to 9×10^{-6}. Accuracy is reported on the test dataset for each training iteration.

Schemes: We compare the following schemes:

- Naive Uncoded: Each client computes a gradient over its local mini-batch selected sequentially, and the server waits to aggregate local gradients from all the clients.
- Greedy Uncoded: Clients compute gradients over their local mini-batches, and the server waits for results from the first $(1-\psi)N$ clients. This corresponds to an aggregate return of $(1-\psi)m$ from the clients.
- CodedFedL: We simulate our approach described in Section III. Client $j \in [n]$ computes gradient over a fixed subset of $\ell_j^*(t^*)$ data points in the local mini-batch, and the server only waits till the deadline time t^* before carrying out the coded federated aggregation in (30) corresponding to the global mini-batch for that iteration. We also include the overhead time for uploading the local parity datasets from the clients to the server.

B. Results

Fig. 4 illustrates the results for MNIST, while Fig. 5 illustrates the results for Fashion MNIST. In Fig. 4(a) and Fig. 5(a), we present the generalization accuracy as a function of wall-clock time for naive uncoded and CodedFedL with different coding redundancy. Clearly, as the coding redundancy is increased by increasing δ, the overall training time reduces significantly. Additionally, as highlighted by the inner figures in Fig. 4(a) and Fig. 5(a), the initial time spent in uploading the coded data to the server generally increases with increased coding redundancy. However, the gain in training time accumulates across training iterations and the impact of this overhead becomes negligible. Furthermore, Fig. 4(a) and Fig. 5(a) illustrate that for the same number of training iterations, CodedFedL achieves a similar generalization accuracy as the naive uncoded scheme even over a large range of coding redundancy. These plots complement our proof of the stochastic approximation of the naive uncoded gradient aggregation by the coded federated gradient aggregation in Section III-E, and show that for even a large coding redundancy, accuracy does not drop significantly.[2]

To highlight the superior performance of CodedFedL when data is non-IID, we compare the convergence plots of generalization accuracy vs training iteration for CodedFedL with $\delta \in \{0.1, 0.2\}$ and greedy uncoded with $\psi \in \{0.1, 0.2\}$. By design of the simulation setting, $\psi = 0.1$ implies that for greedy uncoded, the server misses all the updates associated with a particular class in most iterations, and similarly,

[2]Using the generic local minimizer function fminbnd in MATLAB for solving the concave maximization subproblems, it takes lesser than 2 minutes for implementing our two-step approach for obtaining optimal load allocation and deadline time for all values of coding redundancy considered in the simulations. Although it is negligible compared to the overall training time (see Fig. 4(c) for example), it can be further improved through an implementation specialized for convex programming.

TABLE II
SUMMARY OF RESULTS FOR $\delta = \psi = 0.1$

Dataset	γ (%)	t_γ^U (h)	t_γ^G (h)	t_γ^C (h)	t_γ^U/t_γ^C	t_γ^G/t_γ^C
MNIST	93.3	501	—	198	2.5×	—
	87.5	63	233	27	2.3×	8.8×
Fashion	82.8	521	—	219	2.4×	—
MNIST	82.1	377	224	145	2.6×	1.6×

TABLE III
SUMMARY OF RESULTS FOR $\delta = \psi = 0.2$

Dataset	γ (%)	t_γ^U (h)	t_γ^G (h)	t_γ^C (h)	t_γ^U/t_γ^C	t_γ^G/t_γ^C
MNIST	93.3	501	—	93.2	5.4×	—
	80.2	15.8	125	8.17	1.9×	15×
Fashion	82.8	521	—	90.4	5.8×	—
MNIST	73.8	30.6	123	11.1	2.7×	11×

$\psi = 0.2$ implies that the server misses all the updates associated with two classes in most iterations. As shown in Fig. 4(b) and Fig. 5(b), this results in a poorer generalization performance with respect to training iteration for greedy uncoded in comparison to CodedFedL. Additionally, due to optimal load allocation, CodedFedL performs significantly better than greedy uncoded in the overall training time for identical number of training iterations, as shown in Fig. 4(c) and Fig. 5(c).

Clearly, CodedFedL has significantly better convergence time than the naive uncoded and greedy uncoded approaches, and as highlighted in Section III-E, the coded federated gradient aggregation approximates the naive uncoded gradient aggregation well for large datasets. For further insight, let γ be the target accuracy for a dataset, while t_γ^U, t_γ^G and t_γ^C respectively be the first time instants to reach the γ accuracy for naive uncoded, greedy uncoded and CodedFedL. In Table II, we summarize the results where $\delta = \psi = 0.1$. Gains in the overall training time for CodedFedL are up to $2.5\times$ and $8.8\times$ over naive uncoded and greedy uncoded respectively. In Table III, we compare results for $\delta = \psi = 0.2$, where the gains in the training time for the target accuracy are up to $5.4\times$ and $15\times$ over naive uncoded and greedy uncoded respectively. Also, greedy uncoded never reaches the target accuracy of 93.3% for MNIST and 82.8% for Fashion MNIST, hence the corresponding fields in Tables II and III are empty.

VI. CONCLUSION

We propose CodedFedL, which is the first coding theoretic framework for straggler resilient federated learning in multi-access edge computing networks with general non-linear regression and classification tasks and non-IID data across clients. As a key component, we propose distributed kernel embedding of raw features at clients using a common pseudo-random seed across clients so that they can obtain the kernel features without having to collaborate with each other. In addition to transforming the non-linear federated learning problem into computationally favourable distributed linear regression, kernel embedding enables our novel distributed encoding strategy that generates global parity data for straggler mitigation. The parity data allows the central server to perform gradient computations that substitute or replace missing gradient updates from straggling client devices, thus clipping the tail behavior of gradient aggregation and significantly improving the convergence performance when data is non-IID. Furthermore, there is no decoding of partial gradients required at the central server. We provide an analytical solution for load allocation and coding redundancy computation for obtaining the optimal deadline time, by utilizing statistical knowledge of compute and communication delays of the MEC nodes. Additionally, we provide privacy analysis of generating local parity datasets, and analyze convergence performance of CodedFedL under simplifying assumptions. Finally, we provide results from numerical experiments over benchmark datasets and practical network parameters that demonstrate gains of up to $15\times$ in the wall-clock training time over benchmark schemes.

CodedFedL opens up many interesting future directions. As the global parity dataset is obtained by the MEC server by aggregating the local parity datasets from the clients, the encoded data of each client can be further anonymized by using secure aggregation [53], so that the server gets to know only the global parity dataset, without knowing any individual local parity dataset. With respect to any given client $j \in [n]$, the server will thus receive the sum of the local parity dataset from client j and a noise term, where the noise term will be the sum of the local parity datasets of the remaining clients. Exploring and characterizing this aspect of privacy is left for future study. Furthermore, the problem of characterizing the complete impact of the encoding matrix on convergence and optimizing the deadline time based on convergence criteria will be addressed in future work. Additionally, formulating and studying the load optimization problem based on outage probability for aggregate return is an interesting future work. Adapting CodedFedL to scenarios when the datasets at the clients are changing over time is a motivating future direction as well. Moreover, establishing theoretical foundations of combining coding with random Fourier feature mapping is of significant interest. Another important extension of CodedFedL is to develop coded computing solutions for federated learning for neural network workloads.

APPENDIX A
PROOF OF OPTIMALITY OF THE TWO-STEP APPROACH

Let $(t^*, u^*(t^*), \boldsymbol{\ell}^*(t^*))$ be an optimal solution of (27). Then, $t^* \geq 0$, $0 \leq u^*(t^*) \leq u^{max}$, $\mathbf{0} \leq \boldsymbol{\ell}^*(t^*) \leq (\ell_1, \ldots, \ell_n)$, and the following holds:

$$\begin{aligned}\mathbb{E}(R(t^*; (u^*(t^*), \boldsymbol{\ell}^*(t^*)))) &= m \\ &= \mathbb{E}(R(t^{opt}; (u^{opt}, \boldsymbol{\ell}^{opt}))). \quad (37)\end{aligned}$$

Thus, $(t^*, u^*(t^*), \boldsymbol{\ell}^*(t^*))$ is a feasible solution of the optimization problem in (23). Therefore, we only need to show that $t^* = t^{opt}$. As optimization problem in (23) has a larger solution space than the two-step optimization problem in (27), we have the following inequality:

$$t^{opt} \leq t^* \quad (38)$$

Next, we prove that the optimal expected total aggregate return $\mathbb{E}(R(t;(u^*(t),\boldsymbol{\ell}^*(t))))$ for $t = t^{opt}$ is same as for $t = t^*$, i.e. $\mathbb{E}(R(t^{opt};(u^*(t^{opt}),\boldsymbol{\ell}^*(t^{opt})))) = m$. We first observe that as $(u^*(t),\boldsymbol{\ell}^*(t))$ maximizes the expected total aggregate return for a given deadline time t, we have the following:

$$\begin{aligned}\mathbb{E}(R(t^{opt};(u^*(t^{opt}),\boldsymbol{\ell}^*(t^{opt})))) &\geq \mathbb{E}(R(t^{opt};(u^{opt},\boldsymbol{\ell}^{opt})))\\ &= m \end{aligned} \tag{39}$$

Next, assume that (39) holds with strict inequality. Therefore, by (37), we have the following:

$$\mathbb{E}(R(t^{opt};(u^*(t^{opt}),\boldsymbol{\ell}^*(t^{opt})))) > \mathbb{E}(R(t^*;(u^*(t^*),\boldsymbol{\ell}^*(t^*)))). \tag{40}$$

By the monotonicity of $\mathbb{E}(R(t;(u^*(t),\boldsymbol{\ell}^*(t))))$ with respect to t, (40) implies $t^{opt}>t^*$, which is a contradiction. Hence, $\mathbb{E}(R(t^{opt};(u^*(t^{opt}),\boldsymbol{\ell}^*(t^{opt})))) = m$. Therefore, using the fact that $t = t^*$ is the minimum t such that $\mathbb{E}(R(t;(u^*(t),\boldsymbol{\ell}^*(t)))) = m$, we have $t^*\leq t^{opt}$. Hence, together with (38), the claim $t^* = t^{opt}$ is proved.

APPENDIX B
PROOF OF THEOREM

Using the computation and communication models presented in (11) and (13) in Section II-B, we have the following for the execution time for one epoch for node $j\in[n+1]$:

$$\begin{aligned}T_j &= T_{cmp}^{(j,1)} + T_{cmp}^{(j,2)} + T_{com-d}^{(j)} + T_{com-u}^{(j)}\\ &= \frac{\widetilde{\ell}_j}{\mu_j} + T_{cmp}^{(j,2)} + \tau_j N_{com}^{(j)},\end{aligned} \tag{41}$$

where $N_{com}^{(j)}\sim$NB$(r=2,p=1-p_j)$ has negative binomial distribution while $T_{cmp}^{(j,2)}\sim$E$\left(\frac{\alpha_j\mu_j}{\widetilde{\ell}_j}\right)$ has exponential distribution. Here, we have used the fact that $T_{com-d}^{(j)}$ and $T_{com-u}^{(j)}$ are IID geometric G(p) random variables and sum of r IID G(p) is NB(r,p). Therefore, the probability distribution for T_j is obtained as follows:

$$\begin{aligned}\mathbb{P}(T_j \leq t) &= \mathbb{P}\left(\frac{\widetilde{\ell}_j}{\mu_j} + T_{cmp}^{(j,2)} + \tau_j N_{com}^{(j)} \leq t\right)\\ &= \sum_{\nu=2}^{\infty}\mathbb{P}(N_{com}^{(j)}=\nu)\\ &\quad\cdot\mathbb{P}\left(T_{cmp}^{(j,2)} \leq t - \frac{\widetilde{\ell}_j}{\mu_j} - \tau_j N_{com}^{(j)} | N_{com}^{(j)}=\nu\right)\\ &\overset{(a)}{=} \sum_{\nu=2}^{\infty}\mathbb{P}(N_{com}^{(j)}=\nu)\,\mathbb{P}\left(T_{cmp}^{(j,2)} \leq t - \frac{\widetilde{\ell}_j}{\mu_j} - \tau_j\nu\right)\\ &\overset{(b)}{=} \sum_{\nu=2}^{\infty} U\left(t - \frac{\widetilde{\ell}_j}{\mu_j} - \tau_j\nu\right)(\nu-1)(1-p_j)^2 p_j^{\nu-2}\\ &\quad\cdot\left(1 - e^{-\frac{\alpha_j\mu_j}{\widetilde{\ell}_j}\left(t-\frac{\widetilde{\ell}_j}{\mu_j}-\tau_j\nu\right)}\right),\end{aligned} \tag{42}$$

where (a) holds due to independence of $T_{cmp}^{(j,2)}$ and $N_{com}^{(j)}$, while in (b), we have used $U(\cdot)$ to denote the unit step function with $U(x)=1$ for $x>0$ and $U(x)=0$ for $x\leq 0$. For a fixed t, $\mathbb{P}(T_j\leq t)=0$ if $t\leq 2\tau_j$. For $t>2\tau_j$, let $\nu_m\geq 2$ satisfy the following criteria:

$$(t-\tau_j\nu_m)>0, (t-\tau_j(\nu_m+1))\leq 0. \tag{43}$$

Therefore, for $\nu>\nu_m$, the terms in (b) are 0. Finally, as $\mathbb{E}(R_j(t;\widetilde{\ell}_j)) = \widetilde{\ell}_j\mathbb{E}(\mathbb{1}_{\{T_j\leq t\}}) = \widetilde{\ell}_j\mathbb{P}(T_j\leq t)$, we arrive at the result of our Theorem.

APPENDIX C
PROOF OF MONOTONICALLY INCREASING BEHAVIOR OF OPTIMIZED EXPECTED RETURN

Recall from Section IV that for a given deadline time of t at the server, the expectation of the return $R_j(t;\widetilde{\ell}_j) = \widetilde{\ell}_j\mathbb{1}_{\{T_j\leq t\}}$ for node $j\in[n+1]$ satisfies the following:

$$\mathbb{E}(R_j(t;\widetilde{\ell}_j)) = \begin{cases}\displaystyle\sum_{\nu=2}^{\nu_m} U\left(t-\frac{\widetilde{\ell}_j}{\mu_j}-\tau_j\nu\right)h_\nu f_\nu(t;\widetilde{\ell}_j) & \text{if } \nu_m\geq 2\\ 0 & \text{otherwise}\end{cases}$$

where $U(\cdot)$ is the unit step function with $U(x)=1$ for $x>0$ and 0 otherwise,

$$f_\nu(t;\widetilde{\ell}_j) = \widetilde{\ell}_j\left(1-e^{-\frac{\alpha_j\mu_j}{\widetilde{\ell}_j}(t-\frac{\widetilde{\ell}_j}{\mu_j}-\tau_j\nu)}\right),$$
$$h_\nu = (\nu-1)(1-p_j)^2p_j^{\nu-2},$$

and $\nu_m\in\mathbb{Z}$ satisfies $t-\tau_j\nu_m>0$, $t-\tau_j(\nu_m+1)\leq 0$.

Fix $j\in[n+1]$, and consider a fixed load $\widetilde{\ell}_j$ and a given $\nu\in\mathbb{Z}$. Then, $f_\nu(t;\widetilde{\ell}_j)$ is monotonically increasing in t as $\frac{\partial f_\nu(t;\widetilde{\ell}_j)}{\partial t} = \alpha_j\mu_j e^{-\frac{\alpha_j\mu_j}{\widetilde{\ell}_j}(t-\frac{\widetilde{\ell}_j}{\mu_j}-\tau_j\nu)}>0$ for all $t>0$. Furthermore, by definition, ν_m is monotonically increasing in deadline time t. Therefore, total number of terms in the expression for $\mathbb{E}(R_j(t;\widetilde{\ell}_j))$ increases monotonically with t, and each of those terms increases monotonically with t. Thus, for a fixed $\widetilde{\ell}_j$, $\mathbb{E}(R_j(t;\widetilde{\ell}_j))$ is monotonically increasing in t.

Consider two different deadline times $t=t_1$ and $t=t_2$ with $t_2>t_1$. Based on the discussion in Section IV, let $\widetilde{\ell}_j=\ell_j^*(t_1)$ be the optimal load that maximizes $\mathbb{E}(R_j(t_1;\widetilde{\ell}_j))$ and let $\mathbb{E}(R_j(t_1;\ell_j^*(t_1)))$ be the corresponding optimized expected return. Similarly, let $\widetilde{\ell}_j=\ell_j^*(t_2)$ be the optimal load that maximizes $\mathbb{E}(R_j(t_2;\widetilde{\ell}_j))$ and let $\mathbb{E}(R_j(t_2;\ell_j^*(t_2)))$ be the corresponding optimized expected return. Since $t_2>t_1$ and expected return is monotonically increasing with t, we have $\mathbb{E}(R_j(t_2;\ell_j^*(t_1)))\geq\mathbb{E}(R_j(t_1;\ell_j^*(t_1)))$. Since $\mathbb{E}(R_j(t_2;\ell_j^*(t_2)))$ is the optimal expected return for t_2, it follows that $\mathbb{E}(R_j(t_2;\ell_j^*(t_2)))\geq\mathbb{E}(R_j(t_2;\ell_j^*(t_1)))\geq\mathbb{E}(R_j(t_1;\ell_j^*(t_1)))$. Therefore, the optimized expected return $\mathbb{E}(R_j(t;\ell_j^*(t)))$ is monotonically increasing in the deadline time t.

APPENDIX D
ONE-SHOT SOLUTION FOR AWGN

For node $j\in[n+1]$, consider the function $f_\nu(t;\widetilde{\ell}_j) = \widetilde{\ell}_j(1-e^{-\frac{\alpha_j\mu_j}{\widetilde{\ell}_j}(t-\frac{\widetilde{\ell}_j}{\mu_j}-\tau_j\nu)})$ for $\nu\in\{2,\ldots,\nu_m\}$. We recall that $f_\nu(t;\widetilde{\ell}_j)$ is strictly concave for $\widetilde{\ell}_j>0$. Furthermore, $f_\nu(t;\widetilde{\ell}_j)\leq 0$

for $\widetilde{\ell}_j \geq \mu_j(t-\tau_j\nu)$. Solving for $f'_\nu(t;\widetilde{\ell}_j)=0$, we obtain the optimal load maximizing $f_\nu(t;\widetilde{\ell}_j)$ as follows:

$$\ell^*_j(t,\nu) = -\frac{\alpha_j\mu_j}{W_{-1}(-e^{-(1+\alpha_j)})+1}(t-\nu\tau_j), \tag{44}$$

where $W_{-1}(\cdot)$ is the minor branch of Lambert W-function, where the Lambert W-function is the inverse function of $f(W)=We^W$.

Next, consider the special case of AWGN channel in Section IV, for which the expected return for node $j\in[n+1]$ simplifies as follows:

$$\mathbb{E}(R_j(t;\widetilde{\ell}_j)) = U\left(t-\frac{\widetilde{\ell}_j}{\mu_j}-2\tau_j\right) f_2(t;\widetilde{\ell}_j). \tag{45}$$

When $t\leq 2\tau_j$, $\mathbb{E}(R_j(t;\widetilde{\ell}_j)) = 0$, thus $\ell^*_j(t)=0$. For $t>2\tau_j$, using (44) and the fact that $\widetilde{\ell}_j$ is upper bounded by ℓ_j, $\ell^*_j(t) = \min\{\ell^*_j(t,2),\ell_j\}$, where $\ell^*_j(t,2)$ is as follows:

$$\ell^*_j(t,2) = s_j(t-2\tau_j), \tag{46}$$

where $s_j = -\frac{\alpha_j\mu_j}{W_{-1}\left(-e^{-(1+\alpha_j)}\right)+1}$. As $\ell^*_j(t,2)$ is strictly increasing, there is a unique deadline time for which $\ell^*_j(t,2)=\ell_j$. Let $t=\zeta_j$ be the deadline time for which this holds. Then, using (46), we have the following for ζ_j:

$$\zeta_j = \frac{\ell_j}{s_j}+2\tau_j. \tag{47}$$

Thus, we have the following closed form expression for $\ell^*_j(t)$:

$$\ell^*_j(t) = \begin{cases} 0 & \text{if } t\leq 2\tau_j \\ s_j(t-2\tau_j) & \text{if } 2\tau_j<t\leq\zeta_j \\ \ell_j & \text{otherwise} \end{cases} \tag{48}$$

Furthermore, by (46), the optimal expected return from j-th node for deadline time $2\tau_j<t\leq\zeta_j$ can be simplified as follows:

$$\begin{aligned}\mathbb{E}(R_j(t;\ell^*_j(t))) &= \ell^*_j(t)\left(1-e^{-\frac{\alpha_j\mu_j}{\ell^*_j(t)}\left(t-\frac{\ell^*_j(t)}{\mu_j}-2\tau_j\right)}\right), \\ &= s_j(t-2\tau_j)\left(1-e^{-\alpha_j\left(\frac{\mu_j}{s_j}-1\right)}\right), \\ &= \widetilde{s}_j(t-2\tau_j), \end{aligned} \tag{49}$$

where $\widetilde{s}_j = s_j\left(1-e^{-\alpha_j\left(\frac{\mu_j}{s_j}-1\right)}\right)$. Thus, we have the following closed form expression for the optimal expected return:

$$\mathbb{E}(R_j(t;\ell^*_j(t))) = \begin{cases} 0 & \text{if } t\leq 2\tau_j \\ \widetilde{s}_j(t-2\tau_j) & \text{if } 2\tau_j<t\leq\zeta_j \\ \ell_j\left(1-e^{-\frac{\alpha_j\mu_j}{\ell_j}\left(t-\frac{\ell_j}{\mu_j}-2\tau_j\right)}\right) & \text{otherwise} \end{cases} \tag{50}$$

Thus, we have a closed form for the maximum expected total aggregate return from the nodes till deadline time t as follows:

$$\mathbb{E}(R(t;\boldsymbol{\ell}^*(t))) = \sum_{\substack{j\in[n+1]\\ 2\tau_j<t\leq\zeta_j}} \widetilde{s}_j(t-2\tau_j) + \sum_{\substack{j\in[n+1]\\ \zeta_j<t}} \ell_j\left(1-e^{-\frac{\alpha_j\mu_j}{\ell_j}\left(t-\frac{\ell_j}{\mu_j}-2\tau_j\right)}\right), \tag{51}$$

which is monotonically increasing in t.

Appendix E
Towards Convergence Analysis of CodedFedL

For proving convergence of CodedFedL, we consider $u^*(t^*)$ to be large, and make the following assumption for simplification:

$$\frac{\mathbf{G}^T\mathbf{G}}{u^*(t^*)} = \mathbf{I}_m. \tag{52}$$

The key motivation for our assumption is the observation that by weak law of large numbers, in the limit that the coding redundancy $u^*(t^*)\to\infty$, each diagonal entry in $\frac{\mathbf{G}^T\mathbf{G}}{u^*(t^*)}$ converges to 1 in probability, while each non-diagonal entry converges to 0 in probability. Furthermore, as we demonstrate via numerical experiments in Section V, the convergence curve as a function of iteration for CodedFedL significantly overlaps that of the naive uncoded where the server waits to aggregate the results of all the clients. Hence, for simplifying our analysis, we assume in the remaining proof that $\frac{\mathbf{G}^T\mathbf{G}}{u^*(t^*)}=\mathbf{I}_m$. The general analysis will be addressed in future work.

In the following, we list the remaining assumptions in our analysis:

- Assumption 1: Model parameter space $\mathcal{W}\subseteq\mathbb{R}^{q\times c}$ is a closed and convex set.
- Assumption 2: $\sup_{\boldsymbol{\theta}\in\mathcal{W}}\|\boldsymbol{\theta}-\boldsymbol{\theta}^{(0)}\|_F^2\leq R^2$, where $\boldsymbol{\theta}^{(0)}\in\mathcal{W}$ is given.
- Assumption 3: $\|\frac{1}{\ell^*_j(t^*)}\widetilde{\mathbf{X}}^{(j)T}(\widetilde{\mathbf{X}}^{(j)}\boldsymbol{\theta}-\widetilde{\mathbf{Y}}^{(j)})\|_F^2\leq B_j$ for all $\boldsymbol{\theta}\in\mathcal{W}$ for client $j\in[n]$.
- Assumption 4: Max{singular values of $\widehat{\mathbf{X}}^{(j)}$}$\leq L_j$, for client $j\in[n]$.
- Assumption 5: $\mathbb{P}(T_C\leq t^*)=1$, i.e. the gradient over the coded data is available at the MEC server by t^* with probability 1.

Under the above assumptions, for a given model $\boldsymbol{\theta}\in\mathcal{W}$, the stochastic gradient obtained by the MEC server by the deadline time t^* is as follows:

$$\begin{aligned}&\mathbf{g}_M(\boldsymbol{\theta}) \\ &= \frac{1}{m}(\mathbf{g}_C(\boldsymbol{\theta})+\mathbf{g}_U(\boldsymbol{\theta})), \\ &= \frac{1}{m}\Big(\widehat{\mathbf{X}}^T\mathbf{W}^T\mathbf{W}(\widehat{\mathbf{X}}\boldsymbol{\theta}-\mathbf{Y}) \\ &\quad + \sum_{j=1}^{n}\mathbb{1}_{\{T_j\leq t^*\}}\widetilde{\mathbf{X}}^{(j)T}(\widetilde{\mathbf{X}}^{(j)}\boldsymbol{\theta}-\widetilde{\mathbf{Y}}^{(j)})\Big), \\ &= \frac{1}{m}\Big(\sum_{j=1}^{n}\sum_{\substack{k\in[\ell_j]\\ (\widehat{\mathbf{x}}^{(j)}_k,\mathbf{y}^{(j)}_k)\in\widetilde{D}_j}} (1-\mathbb{P}(T_j\leq t^*))\end{aligned}$$

$$\cdot \widehat{\mathbf{x}}_k^{(j)T}(\widehat{\mathbf{x}}_k^{(j)}\boldsymbol{\theta}-\mathbf{y}_k^{(j)}) + \sum_{j=1}^{n}\sum_{\substack{k\in[\ell_j]\\(\widehat{\mathbf{x}}_k^{(j)},\mathbf{y}_k^{(j)})\in\widehat{D}_j\setminus\widetilde{D}_j}} \widehat{\mathbf{x}}_k^{(j)T}(\widehat{\mathbf{x}}_k^{(j)}\boldsymbol{\theta}-\mathbf{y}_k^{(j)}) + \sum_{j=1}^{n}\sum_{\substack{k\in[\ell_j]\\(\widehat{\mathbf{x}}_k^{(j)},\mathbf{y}_k^{(j)})\in\widetilde{D}_j}} \mathbb{1}_{\{T_j\le t^*\}}\widehat{\mathbf{x}}_k^{(j)T}(\widehat{\mathbf{x}}_k^{(j)}\boldsymbol{\theta}-\mathbf{y}_k^{(j)})\Big). \quad (53)$$

Averaging over the stochastic conditions of compute and communication, we have the following for a given $\boldsymbol{\theta}\in\mathcal{W}$:

$$\begin{aligned}
&\mathbb{E}(\mathbf{g}_M(\boldsymbol{\theta}))\\
&= \frac{1}{m}\Big(\sum_{j=1}^{n}\sum_{\substack{k\in[\ell_j]\\(\widehat{\mathbf{x}}_k^{(j)},\mathbf{y}_k^{(j)})\in\widetilde{D}_j}} (1-\mathbb{P}(T_j\le t^*))\widehat{\mathbf{x}}_k^{(j)T}(\widehat{\mathbf{x}}_k^{(j)}\boldsymbol{\theta}-\mathbf{y}_k^{(j)})\\
&\quad+\sum_{j=1}^{n}\sum_{\substack{k\in[\ell_j]\\(\widehat{\mathbf{x}}_k^{(j)},\mathbf{y}_k^{(j)})\in\widehat{D}_j\setminus\widetilde{D}_j}} \widehat{\mathbf{x}}_k^{(j)T}(\widehat{\mathbf{x}}_k^{(j)}\boldsymbol{\theta}-\mathbf{y}_k^{(j)})\\
&\quad+\sum_{j=1}^{n}\sum_{\substack{k\in[\ell_j]\\(\widehat{\mathbf{x}}_k^{(j)},\mathbf{y}_k^{(j)})\in\widetilde{D}_j}} \mathbb{P}(T_j\le t^*)\widehat{\mathbf{x}}_k^{(j)T}(\widehat{\mathbf{x}}_k^{(j)}\boldsymbol{\theta}-\mathbf{y}_k^{(j)})\Big),\\
&= \frac{1}{m}\sum_{j=1}^{n}\sum_{k=1}^{\ell_j}\widehat{\mathbf{x}}_k^{(j)T}(\widehat{\mathbf{x}}_k^{(j)}\boldsymbol{\theta}-\mathbf{y}_k^{(j)}),\\
&= \nabla_{\boldsymbol{\theta}}\Big(\frac{1}{2m}\sum_{j=1}^{n}\|\widehat{\mathbf{X}}^{(j)}\boldsymbol{\theta}-\mathbf{Y}^{(j)}\|_F^2\Big),\\
&= \mathbf{g}(\boldsymbol{\theta}) \qquad (54)
\end{aligned}$$

Thus, the variance of $\mathbf{g}_M(\boldsymbol{\theta})$ for a given $\boldsymbol{\theta}\in\mathcal{W}$ can be bounded as follows:

$$\begin{aligned}
&\mathbb{E}(\|\mathbf{g}_M(\boldsymbol{\theta})-\mathbb{E}(\mathbf{g}_M(\boldsymbol{\theta}))\|_F^2) \qquad (55)\\
&= \mathbb{E}\Big(\Big\|\frac{1}{m}\sum_{j=1}^{n}(\mathbb{1}_{\{T_j\le t^*\}}-\mathbb{P}(T_j\le t^*))\\
&\quad\cdot\sum_{\substack{k\in[\ell_j]\\(\widehat{\mathbf{x}}_k^{(j)},\mathbf{y}_k^{(j)})\in\widetilde{D}_j}} \widehat{\mathbf{x}}_k^{(j)T}(\widehat{\mathbf{x}}_k^{(j)}\boldsymbol{\theta}-\mathbf{y}_k^{(j)})\Big\|_F^2\Big),\\
&\overset{(a)}{=} \frac{1}{m^2}\sum_{j=1}^{n}\mathbb{P}(T_j\le t^*)(1-\mathbb{P}(T_j\le t^*))\\
&\quad\cdot\Big\|\sum_{\substack{k\in[\ell_j]\\(\widehat{\mathbf{x}}_k^{(j)},\mathbf{y}_k^{(j)})\in\widetilde{D}_j}} \widehat{\mathbf{x}}_k^{(j)T}(\widehat{\mathbf{x}}_k^{(j)}\boldsymbol{\theta}-\mathbf{y}_k^{(j)})\Big\|_F^2,\\
&= \frac{1}{m^2}\sum_{j=1}^{n}\mathbb{P}(T_j\le t^*)(1-\mathbb{P}(T_j\le t^*)) \qquad (56)\\
&\quad\cdot\|\widetilde{\mathbf{X}}^{(j)T}(\widetilde{\mathbf{X}}^{(j)}\boldsymbol{\theta}-\widetilde{\mathbf{Y}}^{(j)})\|_F^2\\
&\le \sum_{j=1}^{n}\frac{(\ell_j^*(t^*))^2}{m^2}\Big\|\frac{1}{\ell_j^*(t^*)}\widetilde{\mathbf{X}}^{(j)T}(\widetilde{\mathbf{X}}^{(j)}\boldsymbol{\theta}-\widetilde{\mathbf{Y}}^{(j)})\Big\|_F^2 \qquad (57)\\
&\le \sum_{j=1}^{n}B_j = B, \qquad (58)
\end{aligned}$$

where in (a), we have used independence of the events $\mathbb{1}_{\{T_{j_1}\le t^*\}}$ and $\mathbb{1}_{\{T_{j_2}\le t^*\}}$ for distinct clients $j_1\in[n]$ and $j_2\in[n]$. Furthermore, for $\boldsymbol{\theta}_1\in\mathcal{W}$ and $\boldsymbol{\theta}_2\in\mathcal{W}$, we have the following bound for smoothness:

$$\begin{aligned}
&\|\mathbf{g}(\boldsymbol{\theta}_1)-\mathbf{g}(\boldsymbol{\theta}_2)\|_F\\
&= \Big\|\frac{1}{m}\sum_{j=1}^{n}\Big(\widehat{\mathbf{X}}^{(j)T}(\widehat{\mathbf{X}}^{(j)}\boldsymbol{\theta}_1-\widehat{\mathbf{Y}}^{(j)})\\
&\quad-\widehat{\mathbf{X}}^{(j)T}(\widehat{\mathbf{X}}^{(j)}\boldsymbol{\theta}_2-\widehat{\mathbf{Y}}^{(j)})\Big)\Big\|_F,\\
&= \frac{1}{m}\Big\|\sum_{j=1}^{n}\Big(\widehat{\mathbf{X}}^{(j)T}\widehat{\mathbf{X}}^{(j)}(\boldsymbol{\theta}_1-\boldsymbol{\theta}_2)\Big)\Big\|_F,\\
&\le \frac{1}{m}\sum_{j=1}^{n}\Big\|\Big(\widehat{\mathbf{X}}^{(j)T}\widehat{\mathbf{X}}^{(j)}(\boldsymbol{\theta}_1-\boldsymbol{\theta}_2)\Big)\Big\|_F,\\
&\overset{a}{\le} \frac{1}{m}\sum_{j=1}^{n}\|\widehat{\mathbf{X}}^{(j)T}\widehat{\mathbf{X}}^{(j)}\|_2\|\boldsymbol{\theta}_1-\boldsymbol{\theta}_2\|_F,\\
&\le \frac{1}{m}\sum_{j=1}^{n}L_j^2\|\boldsymbol{\theta}_1-\boldsymbol{\theta}_2\|_F = L\|\boldsymbol{\theta}_1-\boldsymbol{\theta}_2\|_F. \qquad (59)
\end{aligned}$$

In (a), $\|A\|_2$ denotes the spectral norm of matrix A. Therefore, by Theorem 2.1 in [54], for a total number of r^{max} iterations and a constant learning rate of $\mu^{(r)}=\frac{1}{L+1/\gamma}$ with $\gamma=\sqrt{\frac{2R^2}{Br^{max}}}$, we have the following result:

$$\begin{aligned}
&\mathbb{E}\Big(\frac{1}{2m}\sum_{j=1}^{n}\|\widehat{\mathbf{X}}^{(j)}\boldsymbol{\theta}^{1:r^{max}}-\mathbf{Y}^{(j)}\|_F^2\Big)\\
&\qquad-\min_{\boldsymbol{\theta}\in\mathcal{W}}\frac{1}{2m}\sum_{j=1}^{n}\|\widehat{\mathbf{X}}^{(j)}\boldsymbol{\theta}-\mathbf{Y}^{(j)}\|_F^2\\
&\le R\sqrt{\frac{2B}{r^{max}}}+\frac{LR^2}{r^{max}}, \qquad (60)
\end{aligned}$$

where $\boldsymbol{\theta}^{1:r^{max}}=\frac{1}{r^{max}}\sum_{r=1}^{r^{max}}\boldsymbol{\theta}^{(r)}$. Hence, for achieving an error less than a given $\epsilon>0$, the iteration complexity of CodedFedL is $r^{max}=\mathcal{O}(R^2\max(\frac{2B}{\epsilon^2},\frac{L}{\epsilon}))$.

APPENDIX F
PRIVACY BUDGET FOR CODEDFEDL

We utilize ϵ-mutual-information differential privacy (MI-DP) metric, as proposed in [14], for finding privacy leakage in CodedFedL. For completeness, we first provide the definition of ϵ-MI-DP (shown to be stronger than the standard (ϵ,δ)-DP metric) as presented in [15].

- ϵ-Mutual-Information Differential Privacy: Let $D^N=(D_1,\ldots,D_N)$ be a database with N entries. D^N returns a query as per a randomized mechanism $Q(\cdot)$. Let D^{-i} be the database with all entries except D_i. Then, the randomized mechanism satisfies ϵ-MI-DP if the following is satisfied:

$$\sup_{i,\mathbb{P}(\mathbf{D}^N)} I(\mathbf{D}_i;Q(\mathbf{D}^N)|\mathbf{D}^{-i})\le\epsilon \text{ bits}, \qquad (61)$$

where the supremum is taken over all distributions on $\mathbf{D}^N$.

Next, leveraging the result for random linear projections in [15], we can calculate the privacy budget required for sharing the local parity dataset $(\check{\mathbf{X}}^{(j)}, \check{\mathbf{Y}}^{(j)})$ for a given client $j \in [n]$. As we aim to preserve the privacy of each entry of $\widehat{\mathbf{X}}^{(j)}$, we need to compute the required privacy budget with respect to the largest diagonal entry of the scaling matrix $\mathbf{W}_j$. Therefore, replacing $\mathbf{W}_j$ by an identity matrix, we equivalently consider the privacy leakage for sharing $(\check{\mathbf{X}}^{(j)} = \mathbf{G}_j \widehat{\mathbf{X}}^{(j)}, \check{\mathbf{Y}}^{(j)} = \mathbf{G}_j \mathbf{Y}^{(j)})$ (see Sections III-B and III-D for details). Furthermore, we assume that the entries of $\mathbf{G}_j$ are drawn independently from a standard normal distribution. Then, based on the result for ϵ-MI-DP from Section III-B of [15], CodedFedL needs to allocate ϵ_j privacy budget for sharing $u^*(t^*)$ number of local parity data $(\check{\mathbf{X}}^{(j)}, \check{\mathbf{Y}}^{(j)})$ to the MEC server, where ϵ_j is given by:

$$\epsilon_j = \frac{1}{2} \log_2 \left(1 + \frac{u^*(t^*)}{f^2(\widehat{\mathbf{X}}^{(j)})}\right), \tag{62}$$

where

$$f(\widehat{\mathbf{X}}^{(j)}) = \min_{k_2 \in [q]} \sqrt{\sum_{k_1=1}^{\ell_j} |\widehat{\mathbf{x}}_{k_1}^{(j)}(k_2)|^2 - \max_{k_3 \in [\ell_j]} |\widehat{\mathbf{x}}_{k_3}^{(j)}(k_2)|^2}.$$

Here, we have used $\widehat{\mathbf{x}}_i^{(j)}(k)$ to denote the value of i-th data point corresponding to the k-th feature in raw database $\widehat{\mathbf{X}}^{(j)}$. Intuitively, when the raw data distribution is concentrated along a small number of features, the value of $f(\widehat{\mathbf{X}}^{(j)})$ is small and a larger privacy budget is required for generating coded data to effectively hide those vulnerable features. In contrast, when raw data distribution is uniform in feature space, very little information is leaked by the parity data generated in CodedFedL.

ACKNOWLEDGMENT

This work was a part of Saurav Prakash's internship projects at Intel. The authors sincerely thank the editor and all the reviewers for their valuable feedback and detailed comments.

REFERENCES

[1] S. Dhakal, S. Prakash, Y. Yona, S. Talwar, and N. Himayat, "Coded federated learning," presented at the Wireless Edge Intell. Workshop, IEEE Globecom, Dec. 2019, pp. 1–6.

[2] S. Prakash, S. Dhakal, M. Akdeniz, A. Salman Avestimehr, and N. Himayat, "Coded computing for federated learning at the edge," presented at the Int. Workshop Federated Learn. User Privacy Data Confidentiality, Conjunct. ICML (FL-ICML), 2020.

[3] V. Saravanan, F. Hussain, and N. Kshirasagar, "Role of big data in Internet of Things networks," in *Handbook Research Big Data IoT*. Hershey, PA, USA: IGI Global, 2019, pp. 273–299.

[4] S. Din and A. Paul, "Smart health monitoring and management system: Toward autonomous wearable sensing for Internet of Things using big data analytics," *Future Gener. Comput. Syst.*, vol. 91, pp. 611–619, Feb. 2019.

[5] S. Shahzadi, M. Iqbal, T. Dagiuklas, and Z. U. Qayyum, "Multi-access edge computing: Open issues, challenges and future perspectives," *J. Cloud Comput.*, vol. 6, no. 1, p. 30, Dec. 2017.

[6] S. Kato and R. Shinkuma, "Priority control in communication networks for accuracy-freshness tradeoff in real-time road-traffic information delivery," *IEEE Access*, vol. 5, pp. 25226–25235, 2017.

[7] H. Guo, J. Liu, and J. Zhang, "Efficient computation offloading for multi-access edge computing in 5G HetNets," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2018, pp. 1–6.

[8] Y. Ai, M. Peng, and K. Zhang, "Edge computing technologies for Internet of Things: A primer," *Digit. Commun. Netw.*, vol. 4, no. 2, pp. 77–86, Apr. 2018.

[9] A. Ndikumana *et al.*, "Joint communication, computation, caching, and control in big data multi-access edge computing," *IEEE Trans. Mobile Comput.*, vol. 19, no. 6, pp. 1359–1374, Jun. 2020.

[10] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Artif. Intell. Statist.*, 2017, pp. 1273–1282.

[11] J. Koneý, H. Brendan McMahan, F. X. Yu, P. Richtárik, A. Theertha Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," 2016, *arXiv:1610.05492*. [Online]. Available: http://arxiv.org/abs/1610.05492

[12] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-IID data," 2018, *arXiv:1806.00582*. [Online]. Available: http://arxiv.org/abs/1806.00582

[13] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *Proc. Adv. Neural Inf. Process. Syst.*, 2008, pp. 1177–1184.

[14] P. Cuff and L. Yu, "Differential privacy as a mutual information constraint," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2016, pp. 43–54.

[15] M. Showkatbakhsh, C. Karakus, and S. Diggavi, "Privacy-utility tradeoff of linear regression under random projections and additive noise," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2018, pp. 186–190.

[16] G. Ananthanarayanan, A. Ghodsi, S. Shenker, and I. Stoica, "Effective straggler mitigation: Attack of the Clones.," in *Proc. NSDI*, vol. 13, 2013, pp. 185–198.

[17] D. Wang, G. Joshi, and G. Wornell, "Efficient task replication for fast response times in parallel computation," in *Proc. ACM Int. Conf. Meas. Model. Comput. Syst.*, 2014, pp. 599–600.

[18] S. Li, M. A. Maddah-Ali, Q. Yu, and A. S. Avestimehr, "A fundamental tradeoff between computation and communication in distributed computing," *IEEE Trans. Inf. Theory*, vol. 64, no. 1, pp. 109–128, Jan. 2018.

[19] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Speeding up distributed machine learning using codes," *IEEE Trans. Inf. Theory*, vol. 64, no. 3, pp. 1514–1529, Mar. 2018.

[20] A. Reisizadeh, S. Prakash, R. Pedarsani, and A. S. Avestimehr, "Coded computation over heterogeneous clusters," *IEEE Trans. Inf. Theory*, vol. 65, no. 7, pp. 4227–4242, Jul. 2019.

[21] R. Tandon, Q. Lei, A. G. Dimakis, and N. Karampatziakis, "Gradient coding: Avoiding stragglers in distributed learning," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 3368–3376.

[22] C. Karakus, Y. Sun, S. Diggavi, and W. Yin, "Straggler mitigation in distributed optimization through data encoding," in *Proc. Adv. Neural Inf. Process. Syst.*, pp. 5440–5448, 2017.

[23] S. Li *et al.*, "Coded computing," *Found. Trends Commun. Inf. Theory*, vol. 17, no. 1, pp. 1–148, 2020.

[24] J. Zhang and O. Simeone, "Improved latency-communication trade-off for map-shuffle-reduce systems with stragglers," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2019, pp. 8172–8176.

[25] Q. Yan, M. Wigger, S. Yang, and X. Tang, "A fundamental storage-communication tradeoff for distributed computing with straggling nodes," *IEEE Trans. Commun.*, early access, Aug. 30, 2020, doi: 10.1109/TCOMM.2020.3020549.

[26] C. Karakus, Y. Sun, S. N. Diggavi, and W. Yin, "Redundancy techniques for straggler mitigation in distributed optimization and learning," *J. Mach. Learn. Res.*, vol. 20, no. 72, pp. 1–47, 2019.

[27] M. Ye and E. Abbe, "Communication-computation efficient gradient coding," in *Proc. 35th Int. Conf. Mach. Learn.*, vol. 80, Jul. 2018, pp. 5610–5619.

[28] S. Dhakal, S. Prakash, Y. Yona, S. Talwar, and N. Himayat, "Coded computing for distributed machine learning in wireless edge network," in *Proc. IEEE 90th Veh. Technol. Conf. (VTC-Fall)*, Sep. 2019, pp. 1–6.

[29] Q. Yu, M. Maddah-Ali, and S. Avestimehr, "Polynomial codes: An optimal design for high-dimensional coded matrix multiplication," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 4403–4413.

[30] N. Raviv, R. Tandon, A. Dimakis, and I. Tamo, "Gradient coding from cyclic MDS codes and expander graphs," in *Proc. ICML*, 2018, pp. 4305–4313.

[31] Z. Charles, D. Papailiopoulos, and J. Ellenberg, "Approximate gradient coding via sparse random graphs," 2017, *arXiv:1711.06771*. [Online]. Available: http://arxiv.org/abs/1711.06771

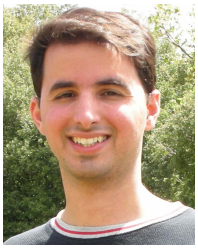
[32] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *Proc. Conf. Mach. Learn. Syst.*, 2020, pp. 429–450.

[33] C. T. Dinh *et al.*, "Federated learning over wireless networks: Convergence analysis and resource allocation," 2019, *arXiv:1910.13067*. [Online]. Available: http://arxiv.org/abs/1910.13067
[34] R. Balakrishnan, M. Akdeniz, S. Dhakal, and N. Himayat, "Resource management and fairness for federated learning over wireless edge networks," in *Proc. IEEE 21st Int. Workshop Signal Process. Adv. Wireless Commun. (SPAWC)*, May 2020, pp. 1–5.
[35] N. Yoshida, T. Nishio, M. Morikura, K. Yamamoto, and R. Yonetani, "Hybrid-FL for wireless networks: Cooperative learning mechanism using non-IID data," in *Proc. ICC - IEEE Int. Conf. Commun. (ICC)*, Jun. 2020, pp. 1–7.
[36] J. D. Jobson, "A multivariate linear regression test for the arbitrage pricing theory," *J. Finance*, vol. 37, no. 4, pp. 1037–1042, Sep. 1982.
[37] T. Isobe, E. D. Feigelson, M. G. Akritas, and G. J. Babu, "Linear regression in astronomy.," *The Astrophys. J.*, vol. 364, pp. 104–113, Dec. 1990.
[38] R. R. Hocking, *Methods and Applications of Linear Models: Regression and the Analysis of Variance*. Hoboken, NJ, USA: Wiley, 2013.
[39] A. Rahimi and B. Recht, "Uniform approximation of functions with random bases," in *Proc. 46th Annu. Allerton Conf. Commun., Control, Comput.*, Sep. 2008, pp. 555–561.
[40] S. Shahrampour, A. Beirami, and V. Tarokh, "On data-dependent random features for improved generalization in supervised learning," in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 1–5.
[41] P. Kar and H. Karnick, "Random feature maps for dot product kernels," in *Proc. Artif. Intell. Statist.*, 2012, pp. 583–591.
[42] J.-H.-R. Chang, A. C. Sankaranarayanan, and B. V. K. V. Kumar, "Random features for sparse signal classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 5404–5412.
[43] *LTE; Evolved Universal Terrestrial Radio Access (e-utra); Physical Channels and Modulation*, document TS 36.211 14.2.0, 3GPP, 2014.
[44] T. A. Courtade and R. D. Wesel, "Optimal allocation of redundancy between packet-level erasure coding and physical-layer channel coding in fading channels," *IEEE Trans. Commun.*, vol. 59, no. 8, pp. 2101–2109, Aug. 2011.
[45] C. Berger, S. Zhou, Y. Wen, P. Willett, and K. Pattipati, "Optimizing joint Erasure- and error-correction coding for wireless packet transmissions," *IEEE Trans. Wireless Commun.*, vol. 7, no. 11, pp. 4586–4595, Nov. 2008.
[46] J.-P. Vert, K. Tsuda, and B. Schölkopf, "A primer on kernel methods," *Kernel Methods Comput. Biol.*, vol. 47, pp. 35–70, Dec. 2004.
[47] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1657–1681, 3rd Quart., 2017.
[48] P. Porambage, J. Okwuibe, M. Liyanage, M. Ylianttila, and T. Taleb, "Survey on multi-access edge computing for Internet of Things realization," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 2961–2991, Dec. 2018.
[49] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
[50] R. M. Corless, G. H. Gonnet, D. E. G. Hare, D. J. Jeffrey, and D. E. Knuth, "On the LambertW function," *Adv. Comput. Math.*, vol. 5, no. 1, pp. 329–359, Dec. 1996.
[51] Y. LeCun, C. Cortes, and C. Burges. (Feb. 2010). *MNIST Handwritten Digit Database*. [Online]. Available: http://yann.lecun.com/exdb/mnist
[52] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," 2017, *arXiv:1708.07747*. [Online]. Available: http://arxiv.org/abs/1708.07747
[53] K. A. Bonawitz *et al.*, "Practical secure aggregation for federated learning on user-held data," in *Proc. NIPS*, 2016, pp. 1–5.
[54] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "QSGD: Communication-efficient SGD via gradient quantization and encoding," in *Adv. Neural Inf. Process. Syst.*, 2017, pp. 1709–1720.

Saurav Prakash (Graduate Student Member, IEEE) received the B.Tech. degree in electrical engineering from IIT Kanpur, India, in 2016. He is currently pursuing the Ph.D. degree in electrical and computer engineering with the University of Southern California (USC), Los Angeles, CA, USA. He was a finalist in the Qualcomm Innovation Fellowship program in 2019. His research interests include information theory and data analytics with applications in large-scale machine learning and edge computing. He received the USC Annenberg Graduate Fellowship in 2016 and was one of the Viterbi-India fellows in Summer 2015.

Sagar Dhakal (Senior Member, IEEE) completed High School in Kathmandu, Nepal. He received the bachelor's degree in electrical and electronics engineering from the Birla Institute of Technology, India, in 2001, and the M.S. and Ph.D. degrees in electrical engineering from the University of New Mexico, Albuquerque, NM, USA, in 2003 and 2006, respectively. He has worked at Intel Laboratories, RIM, Nortel, US Naval Research Laboratory, and a cloud-RAN start-up. He is currently working as a Principal Engineer at Broadcom Corporation, San Jose, CA, USA. His current research interests are in the field of signal processing, wireless communication, and machine learning.

Mustafa Riza Akdeniz (Member, IEEE) received the B.S. degree in electrical and electronics engineering from Bogazici University, Istanbul, Turkey, in 2010, and the Ph.D. degree in electrical and computer engineering from the New York University Tandon School of Engineering, Brooklyn, NY, USA, in 2016. He is currently working as a Research Scientist for Intel Laboratories, Santa Clara, CA, USA. His research interests include wireless channel modeling, information theory, and distributed learning.



Yair Yona received the B.Sc. degree *(magna cum laude)* in electrical engineering, the M.Sc. degree *(magna cum laude)* in electrical engineering, and the Ph.D. degree in electrical engineering from Tel-Aviv University, Israel, in 2005, 2009, and 2014, respectively. He is currently a Staff Engineer at Qualcomm, San Jose, CA, USA.

From 2017 to 2019, he was a Research Scientist at Intel Laboratories. From 2015 to 2017, he was a Post-Doctoral Scholar with the Department of Electrical Engineering, University of California Los Angeles, Los Angeles, CA, USA.

Dr. Yona was a recipient of the Intel Award for excellence in academic studies and research (2009), a Motorola Scholarship in the field of advanced communication (2009), and the Weinstein Prize (2010, 2014) for research in the area of signal processing.

Shilpa Talwar (Member, IEEE) received the M.S. degree in electrical engineering and the Ph.D. degree in applied mathematics from Stanford University in 1996. She is currently the Director of Wireless Multi-Communication Systems with the Intel Laboratories Organization, Intel Corporation. She leads a research team in the Wireless Communications Laboratory, focused on advancements in ultra-dense multi-radio network architectures and associated technology innovations. Her research interests include multi-radio convergence, interference management, mmWave beamforming, and applications of machine learning and artificial intelligence (AI) techniques to wireless networks. While at Intel, she has contributed to IEEE and 3GPP standard bodies, including 802.16m, LTE-advanced, and 5G NR. She is a co-editor of the book on 5G "*Towards 5G: Applications, requirements and candidate technologies*." Prior to Intel, she held several senior technical positions in wireless industry working on a wide-range of projects, including algorithm design for 3G/4G and WLAN chips, satellite communications, GPS, and others. She is the author of 70 technical publications and holds 60 patents.

Salman Avestimehr (Fellow, IEEE) received the B.S. degree in electrical engineering from the Sharif University of Technology in 2003, and the M.S. and Ph.D. degrees in electrical engineering and computer science from the University of California, Berkeley, CA, USA, in 2005 and 2008, respectively.

He is currently a Professor and the Director of the Information Theory and Machine Learning (vITAL) Research Laboratory, Electrical and Computer Engineering Department, University of Southern California. His research interests include information theory, coding theory, and large-scale distributed computing and machine learning. He has received a number of awards for his research, including the James L. Massey Research and Teaching Award from the IEEE Information Theory Society, the Information Theory Society and Communication Society Joint Paper Award, the Presidential Early Career Award for Scientists and Engineers (PECASE) from the White House, the Young Investigator Program (YIP) Award from the U.S. Air Force Office of Scientific Research, the National Science Foundation CAREER Award, the David J. Sakrison Memorial Prize, and several best paper awards at Conferences. He was also the General Co-Chair of the 2020 International Symposium on Information Theory (ISIT). He has been an Associate Editor of the IEEE TRANSACTIONS ON INFORMATION THEORY.

Nageen Himayat (Member, IEEE) received the B.S.E.E. degree from Rice University, the Ph.D. degree from the University of Pennsylvania, and the M.B.A. degree from the Haas School of Business, University of California, Berkeley, CA, USA. She is currently the Director and Principal Engineer with Intel Laboratories, where she conducts research on distributed learning and data centric protocols over 5G/5G+wireless networks. Her research contributions span areas such as machine learning for wireless, millimeter wave and multi-radio heterogeneous networks, cross layer radio resource management, and non-linear signal processing techniques. She has authored over 300 technical publications, contributing to several IEEE peer-reviewed publications, 3GPP/IEEE standards, as well as numerous patent filings. Prior to Intel, she was with Lucent Technologies and General Instrument Corp, where she developed standards and systems for both wireless and wire-line broadband access networks.