To Talk or to Work: Flexible Communication Compression for Energy Efficient Federated Learning over Heterogeneous Mobile Edge Devices

Liang Li*†, Dian Shi‡, Ronghui Hou*†, Hui Li*, Miao Pan‡, and Zhu Han‡
*School of Cyber Engineering, Xidian University, Xi'an, China
†State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences, China
‡Department of Electrical and Computer Engineering, University of Houston, Houston, TX, USA
Email: liliang_1127@outlook.com, dshi3@uh.edu, rhhou@xidian.edu.cn, lihui@mail.xidian.edu.cn, {mpan2, zhan2}@uh.edu

IEEE INFOCOM 2021 - IEEE Conference on Computer Communications | 978-1-6654-0325-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/INFOCOM42981.2021.9488839

***Abstract*—Recent advances in machine learning, wireless communication, and mobile hardware technologies promisingly enable federated learning (FL) over massive mobile edge devices, which opens new horizons for numerous intelligent mobile applications. Despite the potential benefits, FL imposes huge communication and computation burdens on participating devices due to periodical global synchronization and continuous local training, raising great challenges to battery constrained mobile devices. In this work, we target at improving the energy efficiency of FL over mobile edge networks to accommodate heterogeneous participating devices without sacrificing the learning performance. To this end, we develop a convergence-guaranteed FL algorithm enabling flexible communication compression. Guided by the derived convergence bound, we design a compression control scheme to balance the energy consumption of local computing (i.e., "working") and wireless communication (i.e., "talking") from the long-term learning perspective. In particular, the compression parameters are elaborately chosen for FL participants adapting to their computing and communication environments. Extensive simulations are conducted using various datasets to validate our theoretical analysis, and the results also demonstrate the efficacy of the proposed scheme in energy saving.**

***Index Terms*—Federated Learning over Wireless Networks, Gradient Compression, Local SGD, Edge Computing on GPUs.**

I. Introduction

The growing prevalence of mobile smart devices and the rapid advancement of social networking applications result in the phenomenal growth of the data generated at the edge network. To draw useful information from such geographically distributed data, federated learning (FL) has emerged as a promising paradigm that allows participating users to collaboratively learn a shared model, while keeping all the private training data on their edge devices. In particular, all the participants are allowed to run stochastic gradient descent (SGD) locally and send the intermediate gradients to the server periodically for global synchronization. The recent advances in mobile edge computing further facilitate the implementation of FL in mobile networks, since modern smart mobile devices are now armed with high-performance central processing units (CPUs) and graphics processing units (GPUs) to handle intensive computations of intelligent applications. With the technical advantages and implemental feasibilities, FL has seen recent successes in several applications, including next word prediction in Google's Gboard [1], vocal classifier for "Hey Siri" [2], mobile augmented reality [3], etc.

However, to practically deploy FL in wireless networks still faces several critical challenges. On the one hand, both transmitting the gradient updates and performing the local optimizations are resource-hungry, leading to considerable energy consumption at mobile edge devices during the training process. Despite improving computing capacity, mobile edge devices are generally subject to the limited battery lifetime, which hinders their applications in training complex models and supporting continuous learning. On the other hand, the mismatch between the heavy communication loads and the constrained wireless bandwidth hampers the efficient exchange of locally computed updates. The current trend of going deeper in the depth of neural networks has resulted in high-dimensional models with millions of parameters, which inevitably involves significant wireless traffic in global model synchronization. Things can only worsen when considering the heterogeneity of communication environments across different devices, where the learning efficiency may severely depend on a few stragglers with poor channel conditions.

Several pioneering works have been done to manage system resources for efficient FL in wireless networks [4]–[8]. However, these studies overlooked reducing resource consumption intrinsically from the learning algorithm's perspective, hindering a substantial performance boost in resource utilization and training efficiency. A promising solution suggested in recent works in distributed learning is to incorporate state-of-the-art communication compression strategies into FL algorithms, which can considerably reduce the communication cost with little impacts on learning outcomes [9]–[14]. Yet the existing compressed distributed learning algorithms and the corresponding convergence analysis typically require identical compressor across all the participants, which ignores the heterogeneity in participants' communication capacity and thereby exhibits less flexibility. More importantly, this line of works mainly focuses on alleviating communication burdens in FL. However, the ever-increasing deployment of 5G networks that provides data rates as high as 1Gbps has shown

978-1-6654-0325-2/21/$31.00 ©2021 IEEE

a great potential to eliminate the communication bottleneck in FL, let alone the forthcoming 6G revolution [15]. For example, to transmit a Resnet-50, a commonly used deep network for image classification, with approximately 100MB parameters via 1Gbps wireless links typically consumes 0.16J, which is comparable to the energy consumption of performing a single-step local training on one GPU (e.g., 0.2J for NVIDIA Tesla V100 [16], [17]). In light of this fact, it is worthwhile to investigate the impacts of both "working" (i.e., local computing) and "talking" (i.e., wireless transmission) and strike a balance between them via flexible compression control.

In designing a compression control scheme, we would like to communicate as few times and bits as possible to reduce the communication cost. At the same time, we attempt to incur as little distortion to the gradient information as possible towards fast convergence. However, these two goals are fundamentally in conflicts since deeper compression will naturally lead to more distortion on the gradients and more potential communication rounds to converge. When taking energy consumption of edge devices as the measure, this can be further interpreted as follows: Compressing the gradients severely and performing single-step local updates all the time could minimize the energy consumption per communication round. This may require extra communications to attain the targeted model accuracy, or even make the model fail to converge, and thereby impair the overall energy efficiency.

To tackle the challenges above, in this work, we study to improve the energy efficiency of FL over heterogeneous mobile edge devices. Considering the heterogeneous environments across participating wireless edge devices, we propose a flexibly compressed learning algorithm integrating local computation, gradient sparsification, error compensation, and batch size increment. Based on the convergence rate of the algorithm, we develop a compression control scheme that adapts the compression parameters to minimize all the devices' energy consumption on computing and communication. Our salient contributions are summarized as follows:

- We propose a convergence-guaranteed FL algorithm enabling flexible communication compression, which allows participants to compress the gradients to different levels before uploading. The convergence rate is analyzed theoretically and some insightful results are highlighted.
- From the long-term learning perspective, we formulate a compression control problem using the derived convergence bound, where the goal is to achieve energy efficient federated training on edge GPUs over wireless networks.
- Capturing the heterogeneity of participating edge devices in their computing and communication environments, we develop a control algorithm integrating Benders decomposition and inner convex approximation to determine the compression parameters for each participant.
- We evaluate the performance of the proposed control scheme via extensive simulations, which verify the efficacy of our algorithms with various data sources, learning architectures, and system configurations.

The remainder of the paper is organized as follows. Section II reviews related work. Section III elaborates on the flexibly compressed FL procedures and provides the convergence analysis. Section IV presents an energy-efficient compression control algorithm. Section V gives the performance evaluation, and Section VI finally concludes the paper.

II. Related Work

FL over wireless networks has recently gained tremendous attention, whose system design is entangled with training acceleration, network optimization, and on-device resource allocation. Recognizing the limited computing and communication resources at edge computing systems, Wang et al. in [4] dynamically controlled the frequency of global synchronization to minimize the learning loss in real-time adapting to the resource budget. By exploring the unique properties of wireless multiple-access channels, Yang et al. in [5] developed a fast model aggregation approach with joint device selection and beamforming design, which considers only one communication round, and thus cannot guarantee the long-term training performance. To accelerate the training process, Chen et al. in [6] scheduled the participants of high significance for model uploading per communication round while allocating the uplink wireless resources properly. Capturing the trade-off between training time and participants' energy consumption, authors in [7] and [8] formulated optimization problems to jointly allocate the computing and communication resources by considering the heterogeneity of environments. While properly managing the system resources to enable FL in mobile edge networks, these studies overlook reducing resource consumption intrinsically in the essence of learning algorithm itself, thus hindering the substantial boost in training efficiency and resource utilization.

Some recent efforts on distributed learning algorithm design have been devoted to mitigating the communication bottleneck, which can be categorized into two directions: communication round reduction and communication traffic reduction. Specifically, McMahan et al. in [18] proposed the FedAvg algorithm (also known as local SGD) to reduce the frequency of global synchronization, which allows every participant to perform multiple local SGD iterations in a communication round, other than communicating after every local iteration. Authors in [19] and [20] used dynamically increasing batch sizes in distributed SGD to reduce the required number of communication rounds. In [21], a momentum SGD method was adopted to accelerate the convergence where the involved communications during training can be reduced accordingly. To reduce the traffic per communication round, one could let each participant communicate the compressed gradients rather than raw gradients for every global synchronization. For example, sparsified SGD studied in [10], [11] followed the idea that only a small subset of gradients with large magnitude are required to upload. Quantized SGD studied in [12], [14] allowed each participant to quantize the gradients into low-precision values before sending. Despite reduced communication complexity, the huge computing cost remains

hinder FL on resource-constrained edge devices. Besides, most of the current gradient sparsification methods require identical sparsity levels across all the participants, ignoring the heterogeneity of participants and thereby exhibiting less flexibility. This work fills this gap by redesigning the compressed-federated learning algorithm with flexible and well-controlled compression parameters (e.g., global synchronization frequency and gradient sparsity). Simultaneously, heterogeneous computing and communication environments of participants are jointly considered to make the compression strategy suitable for mobile edge networks in practice.

III. Federated Learning with Compression

A. Federated Learning Algorithm Design

We consider an edge computing powered wireless network in which one base station and a set of mobile participants, denoted by $\mathcal{M} = \{1, 2, ..., m, ..., M\}$, collaboratively train a deep neural network model via FL. We follow the common settings of synchronous FL as in [18] and assume that each participant maintains a locally collected dataset. The goal of collaborative training is to learn a global model that achieves uniformly good performance over all the participants, which can be formally described as minimizing a finite-sum non-convex objective $F : \mathbb{R}^d \to \mathbb{R}$ of the form

$$F(\boldsymbol{w}) \triangleq \frac{1}{M} \sum_{m=1}^{M} f_m(\boldsymbol{w}). \tag{1}$$

Here, $f_m(\boldsymbol{w})$ is the loss function defined by the participant m's local dataset $\mathcal{D}_m$ and the parameter vector $\boldsymbol{w}$. Specifically,

$$f_m(\boldsymbol{w}) = \frac{1}{|\mathcal{D}_m|} \sum_{i \in \mathcal{D}_m} f_m^i(\boldsymbol{w}; x_m^i, y_m^i), \tag{2}$$

where $|\mathcal{D}_m|$ is the size of $\mathcal{D}_m$ and (x_m^i, y_m^i) is the i-th sample in $\mathcal{D}_m$. Note here that we usually have $\mathcal{D}_m \neq \mathcal{D}_j$ and $\nabla f_m(\boldsymbol{w}_m) \neq \nabla f_j(\boldsymbol{w}_j)$ for any $m \neq j$ since data are usually heterogeneously-distributed across participants in typical FL applications.

Aiming at reducing the communication cost during FL, we propose the **F**lexible **T**op$_k$ **L**ocal **S**tochastic **G**radient **D**escent with **D**ynamic **B**atch sizes (**FT-LSGD-DB**) algorithm by integrating two state-of-the-art communication compression strategies, namely, local computations and gradient sparsification. The former allows each participant to perform more local computations on the edge device between every two global synchronizations, thereby reducing the total number of communication rounds. The latter lets participants explicitly sparsify the updated gradient tensors before uploading by retaining only a fraction of components, thereby reducing the size of communication payload in each round. Here, we use "Top$_k$" compressor, a commonly used gradient sparsification approach, to take the sparsified top-k gradients. Specifically, for a vector $\boldsymbol{x} \in \mathbb{R}^d$, Top$_k(\boldsymbol{x}) \in \mathbb{R}^d$, and the $i^{th}(i = 1, 2, ..., d)$ element of Top$_k(\boldsymbol{x})$ is defined by:

$$\text{Top}_k(\boldsymbol{x}^{\,i}) = \begin{cases} \boldsymbol{x}^{\,i}, & |\boldsymbol{x}^{\,i}| \geq thr, \\ 0 \ , & \text{otherwise}, \end{cases} \tag{3}$$

Algorithm 1 FT-LSGD-DB Algorithm

Input: The dataset $\{\mathcal{D}_m\}_{\forall m}$; The initialized mini-batch size: $b^{(0)}$; The mini-batch size scaling factor: $\rho > 1$; The number of participants: M; The number of iterations to train: T
Output: Final model parameter $\boldsymbol{w}^{(T)}$
Initialization: $\boldsymbol{w}^{(0)} = \widehat{\boldsymbol{w}}_m^{(0)} = \boldsymbol{e}_m^{(0)}, \ \forall m \in \mathcal{M}$

for $t = 0, 1, 2, ..., T-1$ **do**
  On Edge Devices:
  for $m \in \mathcal{M}$ in parallel **do**
    $b^{(t)} \leftarrow \lfloor \rho^t b^{(0)} \rfloor$
    Sampling a mini-batch $\mathcal{D}_m^{(t)}$ of size $b^{(t)}$ from $\mathcal{D}_m$
    $\widehat{\boldsymbol{w}}_m^{(t+\frac{1}{2})} \leftarrow \widehat{\boldsymbol{w}}_m^{(t)} - \eta^{(t)} \nabla f_m(\boldsymbol{w}^{(t)}; \mathcal{D}_m^{(t)})$
    if $t+1$ is an integer multiple of H **then**
      $\boldsymbol{u}_m^{(t)} \leftarrow \boldsymbol{e}_m^{(t)} + \boldsymbol{w}^{(t)} - \widehat{\boldsymbol{w}}_m^{(t+\frac{1}{2})}$
      $\boldsymbol{g}_m^{(t)} = \text{Top}_{k_m}(\boldsymbol{u}_m^{(t)})$ and upload $\boldsymbol{g}_m^{(t)}$
      $\boldsymbol{e}_m^{(t+1)} \leftarrow \boldsymbol{e}_m^{(t)} + \boldsymbol{w}^{(t)} - \widehat{\boldsymbol{w}}_m^{(t+\frac{1}{2})} - \boldsymbol{g}_m^{(t)}$
      Receive $\boldsymbol{w}^{(t+1)}$ and $\widehat{\boldsymbol{w}}_m^{(t+1)} \leftarrow \boldsymbol{w}^{(t+1)}$
    else
      $\widehat{\boldsymbol{w}}_m^{(t+1)} \leftarrow \widehat{\boldsymbol{w}}_m^{(t+\frac{1}{2})}$
      $\boldsymbol{w}^{(t+1)} \leftarrow \boldsymbol{w}^{(t)}$
      $\boldsymbol{e}^{(t+1)} \leftarrow \boldsymbol{e}^{(t)}$
  At Central Server:
  if $t+1$ is an integer multiple of H **then**
    Collect $\boldsymbol{g}_m^{(t)}, \forall m$ and $\boldsymbol{g}^{(t)} = \frac{1}{M} \sum_{m=1}^{M} \boldsymbol{g}_m^{(t)}$
    $\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} - \boldsymbol{g}^{(t)}$ and broadcast $\boldsymbol{w}^{(t+1)}$
  else
    $\boldsymbol{w}^{(t+1)} \leftarrow \boldsymbol{w}^{(t)}$
return $\boldsymbol{w}^{(T)}$

where $\boldsymbol{x}^{\,i}$ denotes the i^{th} element of $\boldsymbol{x}$ and thr is the k-th largest absolute value of the elements in $\boldsymbol{x}$. In practice, k can be two to three orders of magnitude smaller than d while only sacrificing the model accuracy to a mild extent. In this case, the communication overhead involved in gradient transmission can be dramatically saved [11], [22].

Unlike previous studies requiring identical sparsity level across all the participants, FT-LSGD-DB injects more flexibility into training procedures by allowing the participating devices to perform gradient sparsification with different values of "k". This indeed helps to accommodate stragglers with poor channel conditions and thus mitigates the impacts of stale updates. Besides, FT-LSGD-DB novelly incorporates error compensation and batch size increment into FL procedures, which are two effective methods adopted and verified by practitioners recently. Specifically, error compensation is used to accelerate the global convergence by accumulating the error that arises from only uploading sparse approximations of the gradient updates, which ensures all gradient information does get eventually aggregated [10]. Using dynamically increasing batch sizes during training can maintain the known convergence rate with fewer communication rounds [20]. The pseudocode of our FT-LSGD-DB algorithm is given in Algorithm 1, and the details are described in the following.

Let $\{1, 2, ..., t, ...T\}$ denote a set of iteration indices and

assume that each participant performs H steps of local updates between every two global synchronizations. In each iteration t, evey participant $m \in \mathcal{M}$ performs:

1) *Batch size increment:* Exponentially increases its own SGD batch size with a factor ρ.
2) *Local update:* Update local parameter $\boldsymbol{w}_m$ using the stochastic gradient $\nabla f_m(\boldsymbol{w}; \mathcal{D}_m^{(t)})$, where $\mathcal{D}_m^{(t)}$ is a mini-batch of size b sampled uniformly from $\mathcal{D}_m$ at the t-th iteration.

If aggregation is performed at iteration t (i.e., t is an integer multiple of H), every participant $m \in \mathcal{M}$ performs:

3) *Error compensation:* Add the local error $\boldsymbol{e}_m^{(t)}$ from the previous iteration into the gradient $\boldsymbol{g}_m^{(t)}$.
4) *Gradient sparsification:* Truncate the gradient sum to its top k_m components, sorted in decreasing order of absolute magnitude.
5) *Gradient upload:* Send the sparsified error-compensated gradient $\boldsymbol{g}_m^{(t)}$ to the base station.
6) *Error accumulation:* Update the local error $\boldsymbol{e}_m^{(t)}$.

Upon receiving $\boldsymbol{g}_m^{(t)}$ from all the participants, the base station aggregates them, updates the global model, and broadcasts the new model $\boldsymbol{w}^{(t+1)}$ to participants. Every participant $m \in \mathcal{M}$ set its local parameter $\boldsymbol{w}_m^{(t+1)}$ to be equal to the global parameter $\boldsymbol{w}^{(t+1)}$. The training process above is repeated until achieving satisfactory accuracy. The following section further shows the convergence rate achieved by Algorithm 1 and derives the corresponding communication complexity.

B. Convergence Analysis and Discussion

We consider the following two standard assumptions on the local loss functions $f_m : \mathbb{R}^d \to \mathbb{R}, \forall m \in \mathcal{M}$.

***Assumption** 1 (Smoothness):* $f_m(\cdot)$ is L-smooth, i.e., for every $\boldsymbol{w}, \boldsymbol{w}' \in \mathbb{R}^d$, we have

$$f_m(\boldsymbol{w}) \leq f_m(\boldsymbol{w}') + < \nabla f_m(\boldsymbol{w}), \boldsymbol{w}' - \boldsymbol{w} > + \frac{L}{2}||\boldsymbol{w}' - \boldsymbol{w}||^2. \tag{4}$$

***Assumption** 2 (Bounded variances and second momentum):* For every $\boldsymbol{w}_m^{(t)} \in \mathbb{R}^d$ and $t \in \mathbb{Z}^+$, there exists constants $\sigma > 0$ and $G \geq \sigma$ such that:

$$\mathbb{E}_{\mathcal{D}_m^{(t)} \subset \mathcal{D}_m}[||\nabla f_m(\boldsymbol{w}_m^{(t)}; \mathcal{D}_m^{(t)}) - \nabla f_m(\boldsymbol{w}_m^{(t)}||^2] \leq \sigma^2, \ \forall m, \tag{5}$$

$$\mathbb{E}_{\mathcal{D}_m^{(t)} \subset \mathcal{D}_m}[||\nabla f_m(\boldsymbol{w}_m^{(t)}; \mathcal{D}_m^{(t)})||^2] \leq G^2, \ \forall m. \tag{6}$$

Let $\delta_m = d/k_m \geq 1$ be the gradient sparsity chosen by the m-th participant. Under the assumptions above, the following theorem hold when Algorithm 1 is run with the sparsity series $\{\delta_m\}_{\forall m}$.

***Theorem** 1:* Suppose a constant learning rate $\eta^{(t)} = \eta = \frac{\theta\sqrt{M}}{\sqrt{T}}, \forall t \geq 0$ is chosen where $\theta > 0$ is a constant satisfying $\frac{\theta\sqrt{M}}{\sqrt{T}} \leq \frac{1}{2L}$, we have the convergence rate for Algorithm 1:

$$\begin{aligned} &\mathbb{E}[||\boldsymbol{z}_T||^2] \\ \leq &\frac{4(\mathbb{E}[F(\boldsymbol{w}^{(0)})] - F^*)}{\theta\sqrt{MT}} \\ &+ \frac{8\rho\theta L\sigma^2}{(\rho-1)b^{(0)}\sqrt{M}T^{3/2}} + (4\delta^2+1)\frac{8M\theta^2L^2G^2H^2}{T}, \end{aligned} \tag{7}$$

where $\delta = \sqrt{\frac{1}{M}\sum_{m=1}^{M}\delta_m^2}$ is the root mean square of the sparsity series $\{\delta_m\}_{\forall m}$ and $\boldsymbol{z}_T$ is a random variable which samples a previous parameter $\widehat{\boldsymbol{w}}_m^{(t)}$ with probability $1/MT$.

Proof: Please refer to Appendix[1] for the proof. ■

In Theorem 1, we settle for a weaker notion of convergence and use the average expected squared gradient norm to characterize the convergence rate due to the non-convex settings as [11] does. Based on this, we further give the following corollary on communication complexity of our FT-LSGD-DB algorithm.

***Corollary** 1:* Let $\rho = \frac{T}{T-1}$, $\theta^2 = \frac{b^{(0)}(\mathbb{E}[F(\boldsymbol{w}^{(0)})] - F^*)}{\sigma^2 L}$ and $\mathbb{E}[F(\boldsymbol{w}^{(0)})] - F^* \leq J^2$ where $J \leq \infty$ is a constant. The maximum number of global communication rounds required for achieving an ε-global model convergence, i.e., satisfying $\mathbb{E}[||\boldsymbol{z}_T||^2] \leq \varepsilon$, is given by

$$K(\delta, H) = \mathcal{O}\left(MH\delta^2\right) + \mathcal{O}\left(\frac{1}{\sqrt{M}H}\right). \tag{8}$$

Proof: Substituting $\rho = \frac{T}{T-1}$, $\theta^2 = \frac{b^{(0)}(\mathbb{E}[F(\boldsymbol{w}^{(0)})] - F^*)}{\sigma^2 L}$ and $\mathbb{E}[F(\boldsymbol{w}^{(0)})] - F^* \leq J^2$ into (7) yields

$$\mathbb{E}[||\boldsymbol{z}_T||^2] \leq \frac{12\sigma J\sqrt{L}}{\sqrt{MTb^{(0)}}} + (4\delta^2+1)\frac{8b^{(0)}MLG^2H^2J^2}{\sigma^2 T}. \tag{9}$$

According to the convergence criterion, we suppose that

$$\varepsilon = \frac{12\sigma J\sqrt{L}}{\sqrt{MTb^{(0)}}} + (4\delta^2+1)\frac{8b^{(0)}MLG^2H^2J^2}{\sigma^2 T}. \tag{10}$$

Rearranging the terms, we get the maximum number of iterations as follows:

$$\begin{aligned} T(\delta, H) = &\frac{8b^{(0)}MLG^2J^2H^2(4\delta^2+1)}{\varepsilon\sigma^2} + \frac{72L\sigma^2J^2}{\varepsilon^2 b^{(0)}M} \\ &+ \sqrt{\frac{8b^{(0)}MLG^2J^2H^2(4\delta^2+1)}{\varepsilon\sigma^2} + \frac{36L\sigma^2J^2}{\varepsilon^2 b^{(0)}M}}. \end{aligned} \tag{11}$$

In Algorithm 1, communications are only needed to aggregate individual gradient-update and happen only once every H iterations. Hence, the total number of necessary communication rounds is given by $K = T/H$, i.e.,

$$\begin{aligned} K(\delta, H) = &\frac{8b^{(0)}MLG^2J^2H(4\delta^2+1)}{\varepsilon\sigma^2} + \frac{72L\sigma^2J^2}{\varepsilon^2 b^{(0)}MH} \\ &+ \sqrt{\frac{8b^{(0)}MLG^2J^2(4\delta^2+1)}{\varepsilon\sigma^2} + \frac{36L\sigma^2J^2}{\varepsilon^2 b^{(0)}MH^2}} \\ = &\mathcal{O}\left(MH\delta^2\right) + \mathcal{O}\left(\frac{1}{\sqrt{M}H}\right). \end{aligned} \tag{12}$$

■

The results in (7) and (8) indicate that the gradient sparsity magnitudes of all the participants jointly take impacts on global convergence and communication complexity. Given a target model accuracy (i.e., ε), a higher δ results in a larger bound of communication rounds. Besides, aggressively enlarging H (i.e., "working" more) can also impair the learning efficiency as more communications may be involved.

[1]Appendix is available at https://github.com/anm0/0/blob/master/proof.pdf

IV. Energy-efficient Federated Learning on GPUs: Problem Formulation and Control Algorithm

The theoretical results above reveal that both gradient sparsity levels $\{\delta_m\}_{\forall m}$ and global update frequency H play critical roles in convergence rate and communication efficiency from the learning perspective. Considering a realistic edge computing environment, we highlight that $\{\delta_m\}_{\forall m}$ and H also have great impacts on the energy consumption of participating edge devices, because they affect the payload required for transmitting and the workload required for processing, respectively. In this section, we aim to tune these two types of compression parameters accommodating heterogeneous FL participants for optimizing overall energy efficiency.

A. System Model and Problem Formulation

1) Communication model: Let S_m denote the total number of bits communicated by the m-th participant per global round. Using the "Top_k" compressor defined in (3), one needs to send the values and the positions of the non-zero gradients in the flattened tensors after sparsification. Let FPP denote the floating-point precision, e.g., $\mathrm{FPP} = 32$ for single-precision floating-points and $\mathrm{FPP} = 64$ for double-precision floating-points. With the sparsity $\delta_m = d/k_m$, participant m needs FPP bits to represent the absolute value of each non-zero gradient with one extra bit indicating its sign, i.e.,

$$S_{m,val} = (\mathrm{FPP} + 1) \times k_m \quad \text{bits}. \tag{13}$$

The positions of the non-zero entries can be identified by enumerating all possible sparsity patterns, which require

$$S_{m,pos} = \log_2 \binom{d}{k_m} \quad \text{bits} \tag{14}$$

to represent. Accordingly, we define S_m as

$$S_m = s_1(S_{m,val} + S_{m,pos}) + s_0 \quad \text{bits}, \tag{15}$$

where s_0 and s_1 are coefficients indicating extra communication overhead involved in wireless transmitting [23]. Note that a federated training task usually lasts for a time duration in tens of minutes due to the huge volume of data required for transferring as well as the high computational complexity in running SGD. Thus, the channel conditions of participants may suffer from great fluctuations during a training period. For this reason, it is expected to consider the energy consumption of a training task from a long-term learning perspective. Here, we employ the average transmission rate of every participant $m \in \mathcal{M}$, which is evaluated by

$$R_m = W_m \mathbb{E}_{h_m}[\log_2(1 + \frac{P_m |h_m|^2}{N_0})], \tag{16}$$

where the expectation is taken over channel fading h_m between participant m and the base station; N_0 indicates the power of additive white Gaussian noise; W_m and P_m denote the bandwidth and the transmitting power of participant m, respectively [24]. Afterward, the energy consumed to transmit the sparsified gradients by participant m is calculated as

$$E_m^{com} = \frac{P_m S_m}{R_m}. \tag{17}$$

2) Computational model: On-device learning, especially for training deep network models, is a compute-intensive task that has proved challenging to achieve adequate performance when running merely on CPUs of commodity mobile devices. Fueled by the recent advances in mobile hardware technology, GPU has become a ubiquitous hardware accelerator integrated virtually in every smart device to offer significantly more compute power. A typical GPU chip includes a multi-core GPU module and an associated GPU memory module where the voltage and frequency of GPU cores and GPU memory can be controlled separately. We model the energy consumed to execute a single iteration of GPU-accelerated mini-batch SGD at the m-th edge device as the product of the runtime power and the execution time, i.e.,

$$E_{m,ite}^{cmp} = P_m^{cmp} \cdot T_m^{cmp}, \tag{18}$$

where P_m^{cmp} and T_m^{cmp} are two functions of the core voltage and the core/memory frequency [25], which are given by

$$P_m^{cmp} = P_m^0 + a_1 f_m^{mem} + b_1 (v_m^{core})^2 f_m^{core}, \tag{19}$$

$$T_m^{cmp} = T_m^0 + \frac{a_2}{f_m^{mem}} + \frac{b_2}{f_m^{core}}. \tag{20}$$

Here, P_m^0 and T_m^0 represent the static power consumption and static time consumption; $v_m^{core}, f_m^{core}, f_m^{mem}$ denote the GPU core voltage, GPU core frequency, and GPU memory frequency, respectively; a_1, b_1, a_2 and b_2 are constant coefficients indicating the sensitivity to memory frequency scaling and the core voltage/frequency scaling, which depend on the hardware and the application characteristics. In this work, the value of a_1, b_1, a_2 and b_2 are derived from platform-based experiments by measuring the average runtime energy consumption. Specifically, we measure the energy consumed to execute single-step SGD and estimate the parameters that appeared in the energy model in (18). For simplicity, we shall assume that $E_{m,ite}^{cmp}$ keeps unchanged during training in spite of the incremental batch sizes used in Algorithm 1. This is reasonable due to the fact that GPUs are capable of parallel execution. When the training batch size remains under a threshold, GPUs can process the whole-batch samples simultaneously, leading to a near-constant execution time [16], [26]. In this case, the total energy[2] consumed between every two global synchronizations including H local iterations can be computed as

$$E_m^{cmp} = E_{m,ite}^{cmp} \cdot H. \tag{21}$$

3) Problem Formulation: Given the communication model and the computational model above, we compute the total energy consumption of all the participating edge devices between every two global synchronizations as

$$E = \sum_{m=1}^{M} \left(E_m^{com} + E_m^{cmp} \right), \tag{22}$$

[2]The computational complexity of the gradient sparsification algorithm is so low compared with running local SGD that the corresponding computational workload and the involved energy consumption can be omitted [26].

which captures the heterogeneity of participants on their communication conditions and GPU capacities. Exploiting Corollary 1 and plugging $\delta = \sqrt{\frac{1}{M}\sum_{m=1}^{M}\delta_m^2}$, we model the overall energy consumed during the whole training process as

$$\Gamma(\delta_1, \delta_2, ..., \delta_M, H) = E \cdot \sum_{m=1}^{M} \left(\alpha H \delta_m^2 + \frac{\beta}{M^{3/2} H} \right), \quad (23)$$

where α and β are constants used to approximate the big-$\mathcal{O}$ notion in (8). With the goal of overall energy consumption minimization, we jointly determine the gradient sparsity δ_m for each participant $m \in \mathcal{M}$ and the global update frequency H by solving the following optimization problem:

$$\min_{\{\delta_m, H\}} \quad \Gamma(\delta_1, \delta_2, ..., \delta_M, H) \quad (24a)$$

$$s.t. \quad \delta_{lb} \leq \delta_m \leq \delta_{ub}, \ \forall m, \quad (24b)$$

$$H \in \mathcal{H}. \quad (24c)$$

Here, constraints (24b) and (24c) restrict the feasible range of δ_m and H with $\mathcal{H} \subset \mathbb{Z}^+$, respectively. The formulated problem in (24) exhibits a certain trade-off between compression and convergence in the considered communication-compressed FL setting. To minimize the energy consumption in single global iteration, one will severely compress the gradient tensor and decide to perform a single-step local update all the time, which would greatly impact the convergence rate and increase the number of global synchronizations. As a result, total energy consumption may increase considerably. In practice, the participants with excellent communication environments are expected to adopt slight gradient compression schemes for accelerating the convergence, while the others with poor channel conditions should be allowed to sparsify the gradients more severely to save the energy. In light of this, global update frequency and gradient sparsity should be carefully determined by considering the heterogeneity of participants for achieving energy-efficient FL.

B. Compression Control Algorithm

We develop a compression parameter control algorithm by approximately solving the formulated optimization problem in (24) that falls into the category of mixed-integer nonlinear programming. It is non-trivial to solve since the integer variable H is highly coupled with the continuous variables $\{\delta_m\}_{\forall m}$. In the following, we first transform the permutation operator in (15) into a tractable form. Then we propose an efficient algorithm integrating generalized Benders decomposition and inner convex approximation to find a satisfactory solution of the considered compression control problem.

***Corollary** 2:* Let $\delta_m = d/k_m$ be the chosen gradient sparsity and $\kappa = \mathrm{FPP}+1$. The total number of bits required to be transmitted by participant m per global round, i.e., S_m, can be approximated to

$$S_m(\delta_m) = \frac{s_1 d}{\delta_m}(\log_2 \delta_m + \kappa) + s_0 \quad (25)$$

Proof:

$$\begin{aligned} S_{m,pos} &= \log_2 \binom{d}{k_m} = \log_2 \frac{d!}{(d-k_m)!k_m!} \\ &= \log_2 d! - \log_2 k_m! - \log_2 (d-k_m)! \\ &\overset{(a)}{\approx} d\log_2 d - k_m \log_2 k - (d-k_m)\log_2(d-k_m) \\ &= d\log_2 d - \frac{d}{\delta_m}\log_2\frac{d}{\delta_m} - (d-\frac{d}{\delta_m})\log_2(d-\frac{d}{\delta_m}) \\ &= d[\log_2 \delta_m + (\frac{1}{\delta_m}-1)\log_2(\delta_m - 1)] \overset{(b)}{\approx} \frac{d}{\delta_m}\log_2 \delta_m. \end{aligned}$$

Here, (a) is by Stirling formula that gives precise estimate for factorials, i.e., $\log_2 n! \approx n\log_2 n - n\log_2 e$, and (b) is due to $\delta_m = d/k_m \gg 1$. Substituting $S_{m,pos}$ into (15) yields (25). ■

We can easily verify that $S_m(\delta_m)$ is strongly convex w.r.t. δ_m when $\delta_m \geq e^{3/2}$ by calculating its second order derivative:

$$\frac{\mathrm{d}^2 S_m(\delta_m)}{\mathrm{d}\delta_m^2} = \frac{s_1 d(2\ln\delta_m - 3)}{\delta_m^3 \ln 2} + \frac{2 s_1 d\kappa}{\delta_m^3} > 0. \quad (26)$$

Extensive empirical evidence reveals that $\delta_m \geq e^{3/2}$ always holds in practice so that gradient sparsification can considerably reduce the communication overhead. In the following, we assume that δ_{lb} is set to be no less than $e^{3/2}$ and view $S_m(\delta_m)$ as a strongly convex function without extra conditions. Using Corollary 2, we substitute $\Gamma(\cdot)$ by an approximated energy cost function and rewrite the problem in (24) as:

$$\begin{aligned} \min_{\{\delta_m, H\}} \quad & \sum_{m=1}^{M} \left(\alpha H \delta_m^2 + \frac{\beta}{M^{3/2} H} \right) \\ & \cdot \sum_{m=1}^{M} \left(\frac{P_m s_1 d(\log_2 \delta_m + \kappa)}{R_m \delta_m} + \frac{P_m s_0}{R_m} + E_m^0 H \right) \\ s.t. \quad & (24b) \text{ and } (24c). \end{aligned} \quad (27)$$

To solve (27), we propose an algorithm integrating generalized Benders decomposition and inner convex approximation. Specifically, generalized Benders decomposition performs as the outer-loop algorithm to decompose the problem above into two sub-problems: a primal problem w.r.t. the continuous variables $\{\delta_m\}_{\forall m}$ and a master problem w.r.t. the integer variable H [27]. As the inner-loop algorithm, inner convex approximation is used to solve the primal problem by successively optimizing the approximants of the non-convex objective. We solve the primal problem and the master problem in an alternative and iterative manner, as detailed in Algorithm 2. In each outer-loop iteration, solving the primary problem with given H yields an upper bound for the optimal value of (27) while solving the master problem provides its lower bound. Particularly, we formulate the *primal problem* in the i-th iteration with fixed $H^{(i)}$ as follows:

$$\min_{\{\delta_m\}} \quad \sum_{m=1}^{M} \left(\alpha H^{(i)} \delta_m^2 + \frac{\beta}{M^{3/2} H^{(i)}} \right) \quad (28a)$$

$$\cdot \sum_{m=1}^{M} \left(\frac{P_m s_1 d(\log_2 \delta_m + \kappa)}{R_m \delta_m} + \frac{P_m s_0}{R_m} + E_m^0 H^{(i)} \right)$$

$$s.t. \quad (24b).$$

Algorithm 2 Compression Control Algorithm

Initialization: $H^{(1)} \in \mathcal{H}$; I_{out}; $\epsilon = 10^{-5}$; $\iota = 10^{-5}$; $\xi = 10^{-5}$; $LBD = -\infty$; $UBD = \infty$; $i = 1$;

repeat
  Set inner-loop iteration index $\nu = 0$
  Set step size $\gamma^0 \in (0, 1]$ and start with $\delta_m^0 = \delta_{lb}, \forall m$
  repeat
    Compute $\delta_m^*(\delta_m^\nu), \forall m$ via (35)
    Set $\delta_m^{\nu+1} = \delta_m^\nu + \gamma^0(\delta_m^*(\delta_m^\nu) - \delta_m^\nu), \forall m$
    Set $\nu = \nu + 1$
    Set $\gamma^\nu = \gamma^{\nu-1}(1 - \xi\gamma^{\nu-1})$
  until $||\boldsymbol{\delta}^\nu - \boldsymbol{\delta}^{\nu-1}||_2^2 \leq \iota$
  Save $\delta_m^{(i)} = \delta_m^\nu$ for all $m \in \mathcal{M}$ as the current solution of the primal problem in (28).
  Obtain the Lagrangian multiplier λ and the objective value $\Gamma^{(i)}$
  Update the upper bound value with $UBD = \Gamma^{(i)}$
  Solve the master problem in (29)
  Obtain intermediate solutions of $H^{(i)}$ and $\eta^{(i)}$
  Update the lower bound value with $LBD = \eta^{(i)}$
  Save $H^{(i+1)} = H^{(i)}$
  if $UBD - LBD \leq \epsilon$ **then**
    return The current solutions of $\{\delta_m\}_{\forall m}$ and H.
  else Set $i = i + 1$
until $i = I_{out}$
return The current solutions of $\{\delta_m\}_{\forall m}$ and H.

Note that the primal problem above is always feasible for all $H^{(i)}$ since the continuous variables $\{\delta_m\}_{\forall m}$ are independent of $H^{(i)}$ in the constraint. Thus, we do not need to check the feasibility of the current $H^{(i)}$ as conventional Benders decomposition methods do [28]. Let $\{\lambda_{m1}\}_{\forall m}$ and $\{\lambda_{m2}\}_{\forall m}$ denote two sets of Lagrange multiplier corresponding to the constraints in (24b). We solve the primal problem to get the solutions of $\{\delta_m\}_{\forall m}$, $\{\lambda_{m1}\}_{\forall m}$ and $\{\lambda_{m2}\}_{\forall m}$, which are denoted by $\{\delta_m^{(i)}\}_{\forall m}$, $\{\lambda_{m1}^{(i)}\}_{\forall m}$ and $\{\lambda_{m2}^{(i)}\}_{\forall m}$, respectively. We also update UBD with the objective value of (28a). Afterwards, a feasibility cut can be generated and added to the master problem as a new constraint. In particular, the *master problem* in the i-th iteration is given as follows:

$$\min_{H} \quad \eta \tag{29a}$$

$$s.t. \quad \eta \geq \sum_{m=1}^{M} \left(\alpha H (\delta_m^{(l)})^2 + \frac{\beta}{M^{3/2} H} \right) \cdot \sum_{m=1}^{M} \left(\frac{P_m s_1 d (\log_2 \delta_m^{(l)} + \kappa)}{R_m \delta_m^{(l)}} + \frac{P_m s_0}{R_m} + E_m^0 H \right) + \sum_{m=1}^{M} \left(\lambda_{m1}^{(l)} (\delta_{lb} - \delta_m^{(l)}) + \lambda_{m2}^{(l)} (\delta_m^{(l)} - \delta_{ub}) \right), \forall l = 1, ..., i, \tag{29b}$$

$$H \in \mathcal{H}. \tag{29c}$$

The master problem in (29) is a small-scale mixed-integer programming problem and can be solved using classical optimization algorithms, e.g., Branch-and-Bound. After solving it, we update LBD with the value of η. Now we focus on solving the non-convex primal problem (28), which is specified by the inner-loop procedures in Algorithm 2. Here, we use the inner convex approximation method to find the stationary points iteratively. The main idea is to successively optimize certain approximations of the non-convex objective function in (28a) while maintaining feasibility at each iteration. This requires us to derive a strongly convex approximant of (28a) around each feasible iteration. With a slight abuse of notation, we denote the objective function in (28a) by $\Gamma(\boldsymbol{\delta})$ with $\boldsymbol{\delta} = (\delta_1, ..., \delta_M)$ and rewrite it as the product of two functions, i.e.,

$$\Gamma(\boldsymbol{\delta}) = \Gamma_1(\boldsymbol{\delta}) \cdot \Gamma_2(\boldsymbol{\delta}), \tag{30}$$

where

$$\Gamma_1(\boldsymbol{\delta}) = \sum_{m=1}^{M} \left(\alpha H^{(i)} \delta_m^2 + \frac{\beta}{M^{3/2} H^{(i)}} \right), \tag{31}$$

$$\Gamma_2(\boldsymbol{\delta}) = \sum_{m=1}^{M} \left(\frac{P_m s_1 d (\log_2 \delta_m + \kappa)}{R_m \delta_m} + \frac{P_m s_0}{R_m} + E_m^0 H^{(i)} \right). \tag{32}$$

Note that both $\Gamma_1(\boldsymbol{\delta})$ and $\Gamma_2(\boldsymbol{\delta})$ are positive and strongly convex since non–negative combinations of convex functions preserve convexity. Capturing such the "product of convexity" property of (30), we built an approximation for $\Gamma(\boldsymbol{\delta})$ as:

$$\tilde{\Gamma}(\boldsymbol{\delta}; \boldsymbol{\delta}^\nu) = \Gamma_1(\boldsymbol{\delta}) \cdot \Gamma_2(\boldsymbol{\delta}^\nu) + \Gamma_1(\boldsymbol{\delta}^\nu) \cdot \Gamma_2(\boldsymbol{\delta}), \tag{33}$$

where $\boldsymbol{\delta}^\nu \triangleq (\delta_1^\nu, \delta_2^\nu, ..., \delta_M^\nu)$ denote the current intermediate $\boldsymbol{\delta}$ obtained in the ν-th inner iteration [29]. Obviously, the approximated objective function in (33) is strongly convex and the corresponding approximations of the primal problem in (28) can be solved optimally, which is in the form of:

$$\boldsymbol{\delta}^*(\boldsymbol{\delta}^\nu) = \underset{\{\delta_m\}_{\forall m}}{\operatorname{argmin}} \quad \tilde{\Gamma}(\boldsymbol{\delta}; \boldsymbol{\delta}^\nu) \quad s.t. \quad (24b). \tag{34}$$

***Corollary** 3 (Closed form of $\delta_m^*(\boldsymbol{\delta}^\nu), \forall m$):* Let $\delta_m^*(\boldsymbol{\delta}^\nu)$ be the optimal solution of δ_m given the current $\boldsymbol{\delta}^\nu$. Then, each $\delta_m^*(\boldsymbol{\delta}^\nu)$ has the following expression:

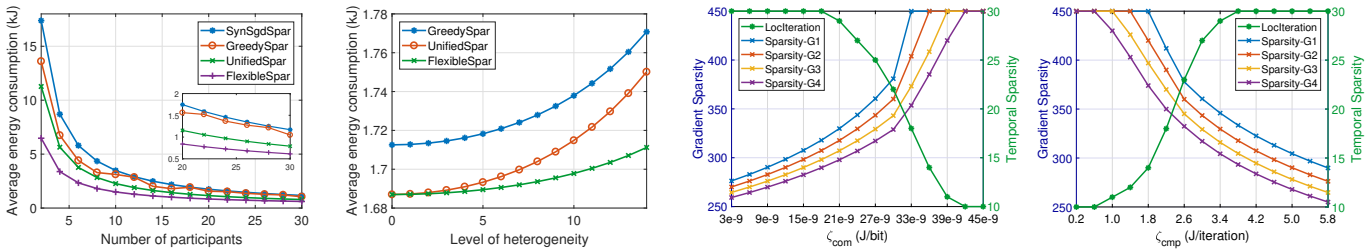
$$\delta_m^*(\boldsymbol{\delta}^\nu) = \begin{cases} \delta_{lb}, & \overline{\delta}_m \leq \delta_{lb} \\ \overline{\delta}_m(\boldsymbol{\delta}^\nu), & \delta_{lb} < \overline{\delta}_m \leq \delta_{ub} \\ \delta_{ub}, & \overline{\delta}_m \geq \delta_{ub} \end{cases} \tag{35}$$

where $\overline{\delta}_m(\boldsymbol{\delta}^\nu)$ is given by

$$\overline{\delta}_m(\boldsymbol{\delta}^\nu) = exp\left(-\frac{1}{3} W\left(-\frac{6AE \cdot exp(3B)}{DC_m} \right) + B \right) \tag{36}$$

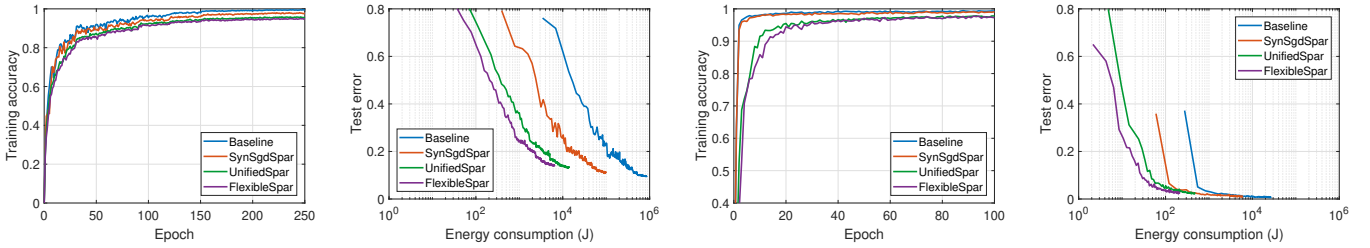
with

$$A = \alpha H^{(i)}, \; B = 1 - \kappa \ln 2, \; C_m = P_m s_1 d / R_m,$$
$$D = \sum_m \left(\alpha H^{(i)} (\delta_m^\nu)^2 + \frac{\beta}{M^{3/2} H^{(i)}} \right),$$
$$E = \sum_m \left(\frac{P_m s_1 d (\log_2 \delta_m^\nu + \kappa)}{R_m \delta_m^\nu} + \frac{P_m s_0}{R_m} + E_m^0 H^{(i)} \right) \ln 2.$$



(a) Impact of system scale (b) Impact of user heterogeneity (c) Impact of comm. capacity (d) Impact of comp. capacity

Fig. 1. Impact of system parameters.



(a) Training accuracy vs epochs (b) Test error vs consumed energy (c) Training accuracy vs epochs (d) Test error vs consumed energy

Fig. 2. Performance on various architectures and datasets. ((a-b): ResNet20 trained on CIFAR-10; (c-d): LeNet5-Caffe trained on MNIST.)

Here, $W(\cdot)$ denotes the Lambert function.

Proof: This is derived by using the Karush-Kuhn-Tucker conditions. We omit the proof due to space limitation. ■

Thanks to the closed-formula for $\delta_m^*(\boldsymbol{\delta}^\nu)$, Algorithm 2 can solve the inner-loop primal problem into optimality without resorting to any iterative solver that can provide approximate solutions only. We adopt the branch-and-bound algorithm to solve the master problem in (29) with computational complexity $O(2^{|\mathcal{H}|})$. When solving the primal problem takes I_{in} inner-loop iterations in total, the overall complexity of Algorithm 2 is $O(I_{out}\max\{O(2^{|\mathcal{H}|}), I_{in}\})$ in the worst case where I_{out} denotes the number of iterations required by the outer loops.

V. Performance Evaluation

We evaluate the performance of the proposed compression control scheme, denoted by "*FlexibleSpars*", via extensive simulations. Particularly, we compare with the following three schemes: 1) *SynSgdSpars* allows participants to sparsify their gradients flexibly, and follows the typical setting of synchronous distributed gradient descent to perform global aggregation after every local update. 2) *GreedySpars* greedily makes the compression control decisions by minimizing the energy cost in the current round, which is oblivious to the impact of the trade-off relationship between the number of global rounds and the cost in a single round on the total energy consumption. 3) *UnifiedSpar* forces every participant to compress the gradients with a unified sparsity, regardless of the heterogeneous communication condition. Compression parameters are determined by solving a simplified version of the problem in (27).

Fig. 1(a) demonstrates the average energy consumption with the varying number of participating devices. We see that the average energy consumption decreases with the growing scale of the FL system under all the schemes, while all the curves tend to be flat. This is due to the fact that increasing the number of participants can help to speed up the convergence of the training process and thereby save the resources at each edge device. Yet such speed-ups will be slight when the participating devices are enough to well capture the whole dataset's information and characteristics $\bigcup_m \mathcal{D}_m$ for training. Among the four schemes, "*SynSgdSpars*" is shown to consume the most energy since it increases the communication complexity significantly. As expected, our proposed scheme "*FlexibleSpar*" outperforms the others as it injects more foresight than "*GreedySpar*" and more flexibility than *UnifiedSpar* into the compression decision-making, to better fit the heterogeneous communication conditions across participants. Fig. 1(b) further gives an insight into the impact of the heterogeneity level of participants' communication capacity on the system energy efficiency. Here, we set the number of participants 12 and divide them into four groups, corresponding to four capacity levels. Assume that participants belonging to the same group use the same wireless bandwidth for gradient exchanges. Let L be the level of heterogeneity that controls the variations in bandwidth among different groups. Fixing the average bandwidth $\overline{W} = 1$GHz, we set the bandwidth adopted by the four groups as $\overline{W} - 0.03L$ (GHz), $\overline{W} - 0.01L$ (GHz), $\overline{W} + 0.01L$ (GHz), and $\overline{W} + 0.03L$ (GHz), respectively. We set the value of L to vary in $\{0, 1, ..., 14\}$, where larger L indicates higher level of heterogeneity. Fig. 1(b) elucidates that the high level of communication heterogeneity has a negative impact on FL and indeed impairs the system energy efficiency.

Notice that we omit the examination of "*SynSgdSpars*" in this setting since it performs so poorly that it is incomparable to the other schemes. Thanks to the flexible compression, our proposed scheme, as we would expect, exhibits more resilience than the others to cope with the scenario with high heterogeneity across participants in terms of wireless channel conditions.

We define $\zeta_{com} = \frac{1}{M}\sum_m \frac{P_m s_1}{R_m}$ (J/bit) and $\zeta_{cmp} = \frac{1}{M}\sum_m E^{cmp}_{m,ite}$ (J/iteration) representing the average energy intensity in terms of transmitting and computing, respectively. Keeping the above group setting of 12 participants with $L = 10$, we further examine the impacts of ζ_{com} and ζ_{cmp} on the optimal values of gradient sparsity $\{\delta\}_{\forall m}$ and synchronization frequency H obtained from our control algorithm. The results are shown in Fig. 1(c)-1(d). Note that H can be viewed as the level of temporal sparsity in the sense that performing multiple local iterations between every two synchronizations implicitly sparsifies the communications in the temporal domain. In Fig. 1(c), "*Sparsity-G1/2/3/4*" denotes the decision of gradient sparsity for the participants in the group 1/2/3/4, respectively. "*LocIteration*" denotes the decision of their temporal sparsity H. When ζ_{com} is small, participants are allowed to sparsify the gradients at a relatively low degree without adding to the total energy cost significantly. In this case, the number of global rounds is dominantly affected by the second term in (8), and a large temporal sparsity H is needed to reduce the number of necessary global rounds. As ζ_{com} grows, all the participants tend to increase the gradient sparsity to fit the worsening communication conditions. We observe that the participants in Group 1 suffering the worst channel conditions generally prefer higher gradient sparsity than the participants in the other groups. When ζ_{com} is large enough, the impact of communication on the total energy consumption becomes more profound than that of computing, forcing the participants to compress the gradients severely with the large $\{\delta_m\}_{\forall m}$ to alleviate the communication burden. Accordingly, the first term in (8) begins to take effects, resulting in the decreasing degree of temporal sparsity H. The analysis above can also be verified by Fig. 1(d), where we vary ζ_{cmp} while keeping ζ_{com} unchanged. We find that the curves in Fig. 1(d) are somewhat symmetrical to the curves in Fig. 1(c). The reason lies in that increasing ζ_{cmp} can be viewed as decreasing ζ_{com} in our system, both of which imply the process of computing cost becoming the bottleneck. Fig. 1(c)-1(d) reveal that there is a trade-off between gradient sparsity and temporal sparsity, which indeed corresponds to less "talking" and less "working". As expected, our flexible compression scheme allows participants to balance these two types of sparsity smoothly against one another for saving energy during training.

Fig. 2 further presents the training results on several commonly used deep models and datasets. Specifically, Fig. 2(a)-2(b) show the convergence rate in terms of epochs and consumed energy respectively for ResNet20 [30] trained on CIFAR-10, while Fig. 2(c)-2(d) show the same things for LeNet5-Caffe [22] trained on MNIST. Here, we take ordinary distributed SGD as the baseline in which each participant transmits full gradients to the server after every local update. From Fig. 2(a) and Fig. 2(c), we observe that "*FlexibleSpar*" exhibits very similar behavior with "*UnifiedSpar*" in terms of convergence rate and final accuracy, both of which slightly underperform "*SynSgdSpar*". This also implies that temporal sparsity has a more profound impact on convergence rate than gradient sparsity in our setting. Due to delayed synchronization and imprecise gradient information, both "*UnifiedSpar*" and "*FlexibleSpar*" are shown to slow down the convergence at initial, which is consistent with our convergence analysis. In spite of this, "*FlexibleSpar*" is validated to be capable of saving energy for on-device training. As reported in Fig. 2(b) and Fig. 2(d), "*FlexibleSpar*" consumes $\times 1.5 - \times 100$ less energy than the other schemes to reach a given target accuracy.

VI. Conclusion

In this work, we have presented a holistic communication compression solution to reduce the energy consumption of FL over heterogeneous participating edge devices without sacrificing the model accuracy. We have developed an FL algorithm enabling flexible communication compression and provided the convergence analysis from a theoretical perspective. Considering the heterogeneous computing and communication conditions across edge devices, we have further designed an energy-efficiency oriented compression control scheme guided by the derived convergence bound. Extensive simulations have been conducted to verify the theoretical analysis and evaluate the algorithm's performance. The results have shown that our flexibly compressed FL scheme exhibits great potentials in accommodating heterogeneous mobile edge devices and improving the energy efficiency of FL over those edge devices.

Acknowledgment

The work of L. Li, R. Hou and H. Li was partially supported by National Natural Science Foundation of China (Grant No. 61571351), State Key Laboratory of Computer Architecture (ICT, CAS) under Grant No. CARCH201904, the Major Research plan of the Shaanxi Science Foundation of China (2019ZDLGY12-08), the 111 project (grant No. B16037), and OPPO funding. The work of D. Shi and M. Pan was supported in part by the U.S. National Science Foundation under grants US CNS-1646607, CNS-1801925, and CNS-2029569. The work of Z. Han was partially supported by NSF EARS-1839818, CNS-1717454, CNS-1731424, and CNS-1702850.

References

[1] A. Hard, K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, "Federated learning for mobile keyboard prediction," *arXiv preprint arXiv:1811.03604*, 2018.

[2] S. Team, "Hey siri: An on-device dnn-powered voice trigger for apple's personal assistant," https://machinelearning.apple.com/research/hey-siri, accessed July, 2020.

[3] D. Chen, L. J. Xie, B. Kim, L. Wang, C. S. Hong, L.-C. Wang, and Z. Han, "Federated learning based mobile edge computing for augmented reality applications," in *Proc. of International Conference on Computing, Networking and Communications (ICNC)*, Big Island, HA, February 2020.

[4] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive federated learning in resource constrained edge computing systems," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1205–1221, 2019.

[5] K. Yang, T. Jiang, Y. Shi, and Z. Ding, "Federated learning via over-the-air computation," *IEEE Transactions on Wireless Communications*, vol. 19, no. 3, pp. 2022–2035, March 2020.

[6] M. Chen, H. V. Poor, W. Saad, and S. Cui, "Convergence time optimization for federated learning over wireless networks," *arXiv preprint arXiv:2001.07845*, 2020.

[7] N. H. Tran, W. Bao, A. Zomaya, N. M. NH, and C. S. Hong, "Federated learning over wireless networks: Optimization model design and analysis," in *Proc. of IEEE Conference on Computer Communications (INFOCOM)*, Paris, France, April 2019.

[8] C. Dinh, N. H. Tran, M. N. Nguyen, C. S. Hong, W. Bao, A. Zomaya, and V. Gramoli, "Federated learning over wireless networks: Convergence analysis and resource allocation," *arXiv preprint arXiv:1910.13067*, 2019.

[9] H. Yu, S. Yang, and S. Zhu, "Parallel restarted SGD with faster convergence and less communication: Demystifying why model averaging works for deep learning," in *Proc. of AAAI Conference on Artificial Intelligence*, Honolulu, HA, January 2019.

[10] S. U. Stich, J.-B. Cordonnier, and M. Jaggi, "Sparsified SGD with memory," in *Proc. of Advances in Neural Information Processing Systems (NIPS)*, Vancouver, Canada, December 2018.

[11] D. Alistarh, T. Hoefler, M. Johansson, N. Konstantinov, S. Khirirat, and C. Renggli, "The convergence of sparsified gradient methods," in *Proc. of Advances in Neural Information Processing Systems (NIPS)*, Vancouver, Canada, December 2018.

[12] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "Qsgd: Communication-efficient SGD via gradient quantization and encoding," in *Proc. of Advances in Neural Information Processing Systems (NIPS)*, Long Beach, CA, December 2017.

[13] J. Ding, G. Liang, J. Bi, and M. Pan, "Differentially private and communication efficient collaborative learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, Virtual Conference, February 2021.

[14] H. Tang, S. Gan, C. Zhang, T. Zhang, and J. Liu, "Communication compression for decentralized training," in *Proc. of Advances in Neural Information Processing Systems (NIPS)*, Vancouver, Canada, December 2018.

[15] W. Saad, M. Bennis, and M. Chen, "A vision of 6g wireless systems: Applications, trends, technologies, and open research problems," *IEEE network*, vol. 34, no. 3, pp. 134–142, February 2019.

[16] T. Lin, S. U. Stich, K. K. Patel, and M. Jaggi, "Don't use large mini-batches, use local SGD," in *Proc. of International Conference on Learning Representations (ICLR)*, Addis Ababa, Ethiopia, April 2020.

[17] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He, "Accurate, large minibatch SGD: Training imagenet in 1 hour," *arXiv preprint arXiv:1706.02677*, 2017.

[18] H. B. McMahan, E. Moore, D. Ramage, S. Hampson *et al.*, "Communication-efficient learning of deep networks from decentralized data," in *Proc. of International Conference on Artificial Intelligence and Statistics (AISTATS)*, Fort Lauderdale, FL, April 2017.

[19] S. L. Smith, P.-J. Kindermans, C. Ying, and Q. V. Le, "Don't decay the learning rate, increase the batch size," in *Proc. of 6th International Conference on Learning Representations (ICLR)*, Vancouver, Canada, April 2017.

[20] H. Yu and R. Jin, "On the computation and communication complexity of parallel SGD with dynamic batch sizes for stochastic non-convex optimization," in *Proc. of 36th International Conference on Machine Learning (ICML)*, Long Beach, CA, June 2019.

[21] H. Yu, R. Jin, and S. Yang, "On the linear speedup analysis of communication efficient momentum SGD for distributed non-convex optimization," in *Proc. of 36th International Conference on Machine Learning (ICML)*, Long Beach, CA, June 2019.

[22] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Sparse binary compression: Towards distributed deep learning with minimal communication," in *2019 International Joint Conference on Neural Networks (IJCNN)*, Budapest, Hungary, July 2019.

[23] S. Khirirat, S. Magnússon, A. Aytekin, and M. Johansson, "Communication efficient sparsification for large scale machine learning," *arXiv:2003.06377*, June 2020.

[24] M. Pan, C. Zhang, P. Li, and Y. Fang, "Joint routing and link scheduling for cognitive radio networks under uncertain spectrum supply," in *Proc. IEEE Conference on Computer Communications (INFOCOM)*, Shanghai, China, April 2011.

[25] X. Mei, X. Chu, H. Liu, Y.-W. Leung, and Z. Li, "Energy efficient real-time task scheduling on cpu-gpu hybrid clusters," in *Proc. of IEEE Conference on Computer Communications (INFOCOM)*, Atlanta, GA, May 2017.

[26] J. Ren, G. Yu, and G. Ding, "Accelerating dnn training in wireless federated edge learning system," *arXiv preprint arXiv:1905.09712*, 2019.

[27] A. M. Geoffrion, "Generalized benders decomposition," *Journal of optimization theory and applications*, vol. 10, no. 4, pp. 237–260, May 1972.

[28] L. Li, D. Shi, R. Hou, R. Chen, B. Lin, and M. Pan, "Energy-efficient proactive caching for adaptive video streaming via data-driven optimization," *IEEE Internet of Things Journal*, vol. 7, no. 6, pp. 5549–5561, March 2020.

[29] G. Scutari, F. Facchinei, and L. Lampariello, "Parallel and distributed methods for constrained nonconvex optimization—part i: Theory," *IEEE Transactions on Signal Processing*, vol. 65, no. 8, pp. 1929–1944, April 2017.

[30] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. of the IEEE conference on computer vision and pattern recognition (CVPR)*, Las Vegas, NV, June 2016.