IEEE INFOCOM 2024 - IEEE Conference on Computer Communications | 979-8-3503-8350-8/24/$31.00 ©2024 IEEE | DOI: 10.1109/INFOCOM52122.2024.10621361

Fed-CVLC: Compressing Federated Learning Communications with Variable-Length Codes

Xiaoxin Su[†], Yipeng Zhou[‡], Laizhong Cui[* †§], John C.S. Lui[¶] and Jiangchuan Liu[‖]
[†]College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China
[‡]School of Computing, Faculty of Science and Engineering, Macquarie University, Sydney, Australia
[§]Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ), Shenzhen University, Shenzhen, China
[¶]Department of Computer Science and Engineering, The Chinese University of Hong Kong, HKSAR
[‖]School of Computing Science, Simon Fraser University, Canada
Email: suxiaoxin2016@163.com, yipeng.zhou@mq.edu.au, cuilz@szu.edu.cn, cslui@cse.cuhk.edu.hk and jcliu@sfu.ca

***Abstract*—In Federated Learning (FL) paradigm, a parameter server (PS) concurrently communicates with distributed participating clients for model collection, update aggregation, and model distribution over multiple rounds, without touching private data owned by individual clients. FL is appealing in preserving data privacy; yet the communication between the PS and scattered clients can be a severe bottleneck. Model compression algorithms, such as quantization and sparsification, have been suggested but they generally assume a fixed code length, which does not reflect the heterogeneity and variability of model updates. In this paper, through both analysis and experiments, we show strong evidences that variable-length is beneficial for compression in FL. We accordingly present Fed-CVLC (Federated Learning Compression with Variable-Length Codes), which fine-tunes the code length in response of the dynamics of model updates. We develop optimal tuning strategy that minimizes the loss function (equivalent to maximizing the model utility) subject to the budget for communication. We further demonstrate that Fed-CVLC is indeed a general compression design that bridges quantization and sparsification, with greater flexibility. Extensive experiments have been conducted with public datasets to demonstrate that Fed-CVLC remarkably outperforms state-of-the-art baselines, improving model utility by 1.50%-5.44%, or shrinking communication traffic by 16.67%-41.61%.**

I. Introduction

Training advanced models, such as Convolutional Neural Network (CNN), needs a massive amount of training data. Collecting the data from clients however may expose their privacy with data breach or privacy infringement risks [1]. To preserve data privacy while training models, the federated learning (FL) paradigm was devised in [2], which orchestrates multiple clients to train a model together without exposing their raw data. FL has attracted tremendous attention from both industry and academia [3] due to its capability in utilizing distributed resources with privacy protection [4].

This work has been partially supported by National Key Research and Development Plan of China under Grant No. 2022YFB3102302, National Natural Science Foundation of China under Grant No. U23B2026 and No. 62372305, Shenzhen Science and Technology Program under Grant No. RCYX20200714114645048, the RGC GRF 14202923 and an NSERC Discovery Grant.

[*]*Corresponding author: Laizhong Cui*

In FL, data samples are owned and distributed on multiple decentralized clients. A parameter server (PS) is deployed to coordinate the model training process for multiple rounds via Internet communications. Briefly speaking, there are three key steps in FL training [2]. In *Step 1*, the PS selects participating clients at the beginning of each communication round, a.k.a., a global iteration, to distribute the latest model; In *Step 2*, each client receiving the model conducts local training iterations with local data samples to update the model; In *Step 3*, model updates will be returned by the participating clients to the PS, which will aggregate the collected model updates to revise the model before ending a communication round. A new communication round will then start from Step 1 again, until terminate conditions are met [5].

Since the distributed clients and the PS communicate over the Internet, the training time of FL can be seriously prolonged in these steps. In particular, in Step 3, the connection between some clients and the remote PS can be quite slow ($\sim$ 1Mbps) [6], and thereby hindering the overall time efficiency. Modern Internet links are often asymmetric, with the uplink capacity being much smaller than the downlink capacity [7], exacerbating the communication bottleneck of FL. These communication challenges are particularly severe when training advanced high-dimensional models, such as ResNet [8] and Transformer [9] with tens of millions of parameters.

To expedite FL, one may compress model updates to be transmitted from participating clients to the PS *with the cost of less accurate model updates*. The compression algorithms for FL can be broadly classified as *quantization* and *sparsification*, which commonly fix the code length. The former expresses each model update with a fewer number of bits. For example, in [10] and [11], each model update is only represented by 2 bits and 1 bit, respectively, to achieve 16 and 32 compression rates by supposing that each original model update takes 32 bits. Different from quantization, sparsification accelerates communications by transmitting a fewer number of model updates. For instance, the Top_k compression algorithm only transmits k model updates of the largest magnitudes from clients to the PS. Top_k can achieve a much higher compression rate than that of quantization in that most model

updates are close to 0 with a very small magnitude in practice [12]. Recently, more sophisticated hybrid compression was proposed [13] by combining quantization and sparsification.

The core problem in FL model compression is how to minimize the accuracy loss of model updates due to compression. Using the fixed code length for model compression fails to exploit the fact revealed by existing works [12], [14] that the distribution of the magnitudes of model updates is very skewed, and thus existing works cannot fully minimize the compression loss. In view of this gap, we propose a novel Federated Learning Compression with Variable-Length Codes (Fed-CVLC) algorithm, which employs codes of different lengths to compress model updates. To motivate our study, we use a simple example to illustrate the idea of variable-length codes for model compression. Then, the optimization problem to minimize the loss function subject to limited communication traffic is formulated with respect to the code length of each model update. Besides, our proposed method is friendly for protocol implementation in that model updates encapsulated into the same communication packet are constrained by the same code length to avoid excessive overhead. Based on our analysis, we show that the formulated problem can be efficiently solved to optimally determine the code length of each model update.

The advantage of our design lies in that Fed-CVLC unifies the design of quantization and sparsification since quantization and sparsification can be regarded as special cases of Fed-CVLC. Fed-CVLC degenerates to quantization compression, if the code length is fixed as a constant for all model updates. Similarly, Fed-CVLC degenerates to the Top_k sparsification compression if we fix the code length for k top model updates and set code length as 0 for remaining model updates. Therefore, quantization and sparsification are rigid in the sense that they only set one or two code lengths for compression. In contrast, Fed-CVLC is more general and flexible by taking the code length as a variable toward compression in FL.

At last, we conduct extensive experiments with CIFAR-10, FEMNIST and CIFAR-100 datasets using Fed-CVLC and state-of-the-art compression algorithms. The experimental results demonstrate the superb performance of our Fed-CVLC algorithm, which can improve model accuracy by 3.21% and shrink communication traffic by 27.64% on average in comparison with state-of-the-art compression baselines.

The rest of the paper is organized as follows. We introduce the related works in Sec. II and preliminaries in Sec. III. The detailed analysis of our problem and the design of Fed-CVLC with a discussion of its implementation are delivered in Sec. IV. Ultimately, we conclude our work in Sec. VI after evaluating the performance of Fed-CVLC in Sec. V.

II. Related Work

In this section, we discuss related works from two perspectives: federated learning design and model compression.

Federated learning (FL) is a novel distributed machine learning framework, which was originally proposed by Google [2] to protect mobile user privacy. FedAvg [2] has become the most fundamental algorithm for conducting FL. FedAvg performs multiple local iterations in each round to reduce communication frequency between clients and the PS. The convergence rate of FedAvg was derived in [15] and [16] to validate the effectiveness of FL in model training. Various variants of FedAvg were devised to improve FL from different perspectives [17]–[19]. In [20], [21], the authors discussed the effect of client sampling and designed client scheduling strategies to optimize the model performance. Wang *et al.* [22] designed a FL training paradigm with hierarchical aggregation and proposed an efficient algorithm to determine the optimal cluster structure with resource constraints.

Model compression is perpendicular to the above works, which significantly improves the training efficiency of FL by largely shrinking communication traffic. We briefly discuss two most popular model compression approaches: quantization and sparsification.

The essence of quantization is to use a fewer number of bits to represent each model update. PQ [23] and QSGD [24] are efficient quantization algorithms that map model updates to a finite set of discrete values in an unbiased manner, thereby reducing the number of bits for each update. DAdaQuant was designed by [25] using time- and client- adaption to adjust quantization levels, optimizing the final model utility of FL.

Sparsification, achieving a much higher compression rate, is more aggressive by removing unimportant model updates from communications. [26] proposed to drop a proportion of model parameters stochastically and scale the remaining parameters. The Top_k algorithm [27], [28] only transmits k model updates of the largest magnitudes. An error compensation mechanism was used to guarantee convergence performance. [13], [29] came up with novel compression algorithms by combining quantization and sparsification. The convergence of these algorithms was theoretically analyzed.

Nonetheless, all these quantization and sparsification algorithms fix the code length for model compression regardless the skewness of the magnitude distribution of model updates. Different from existing works, Fed-CVLC adopts variable-length codes to better gauge the importance of model updates by using more bits to represent larger magnitudes and fewer bits to represent smaller magnitudes.

III. Preliminaries

Consider a generic FL system with $N > 1$ clients in the system. Local data samples on each client i are drawn from a local data distribution denoted by $\mathcal{D}_i$, where $i \in [N]$. The objective of FL is to train a model vector $\mathbf{w} \in \mathbb{R}^d$ that minimizes the global loss function $F(\mathbf{w})$ defined across all N clients [15]. In other words, $\mathbf{w}^* = \arg\min_{\mathbf{w}} \left[F(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N} \mathbb{E}_{\xi\sim\mathcal{D}_i} f(\xi, \mathbf{w})\right]$, where ξ is a particular data sample randomly drawn from the local data distribution $\mathcal{D}_i$ and $f()$ is the loss function evaluating the model using a particular sample.

Let $F_i(\mathbf{w}) = \mathbb{E}_{\xi\sim\mathcal{D}_i} f(\xi, \mathbf{w})$ denote the local loss function on client i. FedAvg [2], the most fundamental model training

algorithm in FL, and its variants [30], [31] commonly have the following three steps for deriving $\mathbf{w}^*$.

1) At the beginning of communication round t, the PS distributes the latest vector $\mathbf{w}_t$ to a number of selected clients, denoted by set $\mathcal{S}_t$.
2) Each selected client i will use downloaded $\mathbf{w}_t$ to initialize her local training vector as $\mathbf{w}^i_{t,0} = \mathbf{w}_t$ and update the model locally for $E \geq 1$ local iterations $\mathbf{w}^i_{t,j+1} = \mathbf{w}^i_{t,j} - \eta_{t,j} \nabla F_i(\mathbf{w}^i_{t,j}, \mathcal{B}^i_{t,j})$ for $j = 0, \ldots, E-1$. Here $\mathcal{B}^i_{t,j}$ is a data sample batch (with size $|\mathcal{B}^i_{t,j}|$) randomly selected from client i's local dataset for the j-th local iteration. Model updates updated by client i in communication round t are denoted by $\mathbf{U}^i_t = \mathbf{w}^i_{t,0} - \mathbf{w}^i_{t,E}$, which will be returned to the PS.
3) The PS receiving $\mathbf{U}^i_t$'s from all participating clients will update the model by aggregating model updates, *i.e.*, $\mathbf{w}_{t+1} = \mathbf{w}_t - \frac{1}{|\mathcal{S}_t|} \sum_{i \in \mathcal{S}_t} \mathbf{U}^i_t$. Then, the PS goes back to Step 1 until total $T \geq 1$ iterations are executed.

To reduce the communication traffic for transmitting $\mathbf{U}^t_i$, various compression algorithms can be applied to compress $\mathbf{U}^t_i$ into $\hat{\mathbf{U}}^t_i$ which can consume much less communication bandwidth. The discrepancy between $\mathbf{U}^t_i$ and $\hat{\mathbf{U}}^t_i$ is called the compression error, a.k.a., the accuracy loss of model updates attributed to model updates compression.

IV. Analysis and Algorithm Design

In this section, we introduce a simple example to illustrate the intuition of our design, and then formulate the problem to minimize the compression error with a fixed compression rate using variable-length codes. At last, we discuss how to implement our algorithm with lightweight overhead.

A. A Motivation Example

State-of-the-art quantization and sparsification compression algorithms generally set the fixed code length to compress model updates, which unfortunately may not minimize the compression error of model updates with different values. For example, suppose we use fixed 6 bits to encode two numbers with values 1,000 and 2, respectively. Encoding the number 1,000 then loses accuracy significantly, whereas encoding the number 2 wastes bits. This observation inspires us to consider encoding model updates with variable-length codes.

We use an example to illustrate the intuition of our algorithm. In Fig. 1, we plot the distribution of model updates by training a CNN model with 300,000 parameters. Here, the x-axis represents the ranked ID of model updates (in a log scale because most model updates are close to 0), the left y-axis represents the magnitude of model updates and the right y-axis represents the number of bits to encode each model update. The continuous curve represents the magnitude of model updates, while each vertical bar represents the number of bits to encode its corresponding model update.

From Fig. 1, we can observe that the distribution of model update magnitudes is very skewed with only a small number of model updates far away from 0. In Fig. 1(a), quantization uniformly compresses each model update with 6 bits regardless of the magnitude. Sparsification plus quantization plotted in Fig. 1(b) is more advanced than quantization by using 6 bits to encode top 3,000 model updates (*i.e.*, top 1%) and 0 bits to encode the rest small model updates. The principle of our algorithm is presented in Fig. 1(c), which is more flexible in setting the code length in accordance with the model update magnitude for compression.

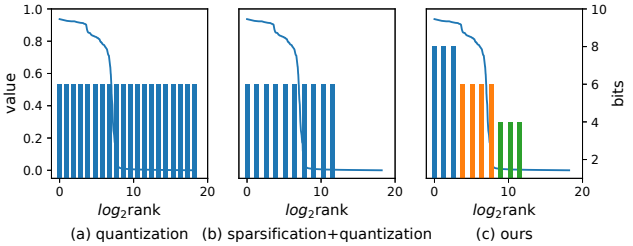

Fig. 1. The distribution of model updates in a particular round and the code length of each model update set by different compression algorithms.

B. Compression Error

The core problem of model compression when shrinking communication traffic lies in incurred compression error causing inaccuracy of model updates. To quantify the impact of a model compression algorithm on model utility, it is imperative to explicitly analyze and quantify the compression error.

To make analysis of compression algorithms formulable, it is common to use a distribution to model the magnitudes of model updates. For example, in [27], it was assumed that the distribution of model update magnitudes is uniform. However, this simple uniform distribution cannot capture the skewness of model updates in practice. In [14], a more generic power law distribution was adopted to better represent the distribution of model update magnitudes.

Definition 1. *On client i, model updates ranked in a descending order of their absolute values satisfy a power law decreasing,* i.e., $|\mathbf{U}^i_t\{l\}| \leq \phi_i l^{\alpha_i} \quad \forall l \in \{1, 2, \ldots, d\}$, *where $\mathbf{U}^i_t\{l\}$ is the l-th largest model update (in terms of absolute value) in $\mathbf{U}^i_t$, $\alpha_i < 0$ is the decay exponent controlling the decaying rate of the distribution and ϕ_i is a constant.*

There are three operations to compress model updates $\mathbf{U}^i_t$.

- **Sparsification.** First, sparsification is applied to remove model updates of a small magnitude yielding $\overline{\mathbf{U}}^i_t$.
- **Quantization.** Second, unbiased quantization is applied to further compress remaining top model updates with a fixed number of bits. $\widetilde{\mathbf{U}}^i_t$ is model updates after unbiased quantization satisfying $\mathbb{E}[\widetilde{\mathbf{U}}^i_t] = \overline{\mathbf{U}}^i_t$.
- **Scaling.** Third, $\widetilde{\mathbf{U}}^i_t$ should be scaled down by a constant B to theoretically guarantee the convergence of FL with compressed model updates. In other words, the scaling operation is $\hat{\mathbf{U}}^i_t = \frac{\widetilde{\mathbf{U}}^i_t}{B}$.

Let $\hat{\mathbf{U}}^i_t$ denote final compressed model updates after three operations. Each operation will incur compression error. The error attributed to quantization and scaling operations is quantified as follows.

Lemma 1. *Suppose a model update vector $\overline{\mathbf{U}}$ with z_y elements and each element is quantified by y bits $\widetilde{\mathbf{U}}$. After quantization and scaling operations, the error between $\overline{\mathbf{U}}$ and $\hat{\mathbf{U}}$ is $\mathbb{E}\|\hat{\mathbf{U}} - \overline{\mathbf{U}}\|^2 \leq \frac{Q(z_y,y)+(B-1)^2}{B^2}\|\overline{\mathbf{U}}\|^2$, where $Q(z_y, y)$ is the quantization error to quantify z_y model updates with each model update expressed by y bits.*

The detailed proof is presented in Appendix A-A. Note that our design is a generic framework in the sense that different unbiased quantization algorithms can be applied in our design. Thus, $Q(z_y, y)$ needs not to be fixed until we specify some quantization algorithms. In general, we can encode top k model updates with Y kinds of codes of different lengths. For example, if $Y = 32$, quantization compression, *e.g.*, QSGD, can compress a model update with $32, 31, \ldots,$ or 1 bit. Intuitively speaking, a model update of a larger magnitude should be quantified with more bits. Thus, we put ranked top model updates into Y groups. Formally, we let $\mathcal{Z}_y$ with cardinality z_y represent the set of model updates that will be compressed with codes of length y.

Considering that model updates will be grouped to compress with different code lengths, the compression error after three operations can be bounded as below.

Proposition 1. *On client i in communication round t, the error of model updates after sparsification operation selecting top k model updates, quantization operation with Y kinds of variable-length codes and scaling operation with factor B is bounded by $\mathbb{E}\|\mathbf{U}_t^i - \hat{\mathbf{U}}_t^i\| \leq \gamma_i \|\mathbf{U}_t^i\|^2$. Here,*

$$\gamma_i = \frac{d^{\beta_i} - (k+1)^{\beta_i}}{d^{\beta_i} - 1} + \frac{1}{d^{\beta_i} - 1}\sum_{y=1}^{Y} \Big(\big(\frac{Q(z_y, y)}{B^2} + \frac{1}{B_c^2}\big)\big(Z_y^{\beta_i} - Z_{y+1}^{\beta_i}\big)\Big), \tag{1}$$

where $\beta_i = 2\alpha_i + 1$, $k = \sum_{y=1}^{Y} z_y$ and $Z_y = z_Y + z_{Y-1} + \cdots + z_y$. $Q(z_y, y)$ is defined in Lemma 1. To guarantee the convergence of FL, it is required that $0 < \gamma_i < 1$. B is set as a constant scale factor satisfying $B > \max_{\forall y} \frac{Q(z_y,y)+1}{2}$, to ensure the convergence of FL [1]. B_c is a constant satisfying $\frac{1}{B} + \frac{1}{B_c} = 1$, which is used to simplify our presentation.

In Proposition 1, γ_i is a critical parameter. A smaller γ_i implies a smaller compression error and hence higher model utility. Proposition 1 is proved by combining the compression error for model updates in Y different groups. The detailed proof is presented in Appendix A-B.

Our problem lies in how to optimally determine the number of model updates z_y that should be encoded into y bits, *e.g.*, $y = 1, \ldots, 32$.

To demonstrate the generic merit of our approach, we list the convergence rates of the biased compression algorithms with compression error γ_i proposed in existing works in Table I. Note that we have ignored all constants that are not asymptotically relevant with the compression error and only show asymptotic convergence rates in Table I. These works derived convergence rates in different expressions because they are slightly different in assumptions of loss functions and setting of the learning rates.

Based on convergence rates listed in Table I, we can conclude that our approach is effective in improving convergence and hence model utility for all these works by lowering the compression error without compromising the compression rate. More specifically, we draw the following conclusions.

- Lowering γ_i for a given traffic limitation can always make the gap to the convergence point smaller, and hence lower the loss function and improve model utility.
- γ_i's are independent with each other implying that each client can independently make her local optimization of γ_i so as to optimize the global model utility.

Based on the above observations, our problem becomes how to locally minimize the compression error γ_i on client i by tuning z_y's. To make our discussion concise, we only discuss how to minimize γ for an arbitrarily selected client hereafter.

C. Optimizing Code Length

Before we formulate the problem to optimize the code length for each model update, we define the constraints of model transmission in FL.

By only transmitting top model updates, it is necessary to transmit the position ID of a selected top model update together with its value in FL. It is well-known that data transmitted via Internet communications will be partitioned and encapsulated into packets. Each packet transmitted in IP based networks has a header and a payload. The header part contains the necessary network information to decode the packet. The payload part contains position IDs and values of model updates. Suppose that there are k top model updates that will be transmitted via R packets. In practice, the size of each packet is limited and denoted by b_r. Let $\mathcal{P}_r$ with cardinality P_r denote the set of model updates that will be encapsulated into packet r where $1 \leq r \leq R$.

Let u denote a particular model update. By slightly abusing our notations, let y_u denote the code length to express model update u. Due to the constraint of the packet size, we have

$$\sum_{u \in \mathcal{P}_r} (s + y_u) + H \leq b_r, \quad \textit{for } r = 1, \ldots, R. \tag{2}$$

Here H is the header size of packets and s is the size of each position ID. If there are total d model updates in $\mathbf{U}_t^i$, we have $s = \lceil \log_2 d \rceil$. Equation (2) indicates that it is impracticable to encapsulate all model updates (for training advanced high-dimensional models) into a single packet due to the limited size of each packet.

From (2), we can observe that the communication efficiency is higher if the header part with size H is smaller relative to the packet size b_r. However, it is rather difficult to straightly minimize the size of the header part because it is dependent on the way how model updates are encapsulated into packets. When using variable-length codes, it is possible that model

[1]The exact value of B is $\max_{\forall y} Q(z_y, y) + 1$, which is difficult for analysis. Thus, in each global iteration we use $\max_{\forall y} Q(z_y, y) + 1$ of the previous iteration for approximation.

TABLE I

CONVERGENCE OF DIFFERENT LOSS FUNCTIONS IN FEDERATED LEARNING USING DIFFERENT LEARNING RATES UNDER COMPRESSED COMMUNICATION: F^* IS THE MINIMUM VALUE OF THE GLOBAL LOSS FUNCTION $F()$, $\mathbf{w}_T$ IS THE AGGREGATION OF MODELS FROM DIFFERENT CLIENTS, $\nabla F(\mathbf{w})$ IS THE GLOBAL GRADIENT OF MODEL $\mathbf{w}$, $\mathbf{v}_1$ IS A RANDOM VARIABLE DRAWN FROM THE SEQUENCE $\{\mathbf{w}_t\}$ WITH THE LEARNING RATE AS THE WEIGHT, $\mathbf{v}_2$ IS A RANDOM VARIABLE DRAWN FROM THE SEQUENCE OF $\{\mathbf{w}_{t,j}^i\}$ UNIFORMLY, ϵ_i IS A CONSTANT SATISFYING $\gamma_i < \epsilon_i < 1$.

Algorithm	Loss Function	Learning Rate	Asymptotic Convergence Rate
Qsparse-local-SGD [29]	strongly convex	$\mathcal{O}(\frac{1}{t})$	$F(\mathbf{w}_T) - F^* \leq \mathcal{O}(\frac{1}{T} + \sum_{i=1}^{N} \frac{\gamma_i(1+\gamma_i)}{T^2})$
CFedAvg [32]	non-convex	$\mathcal{O}(\frac{1}{\sqrt{t}})$	$\|\nabla F(\mathbf{w}_T)\|^2 \leq \mathcal{O}(\sum_{i=1}^{N} \frac{\gamma_i}{(\epsilon_i-\gamma_i)\sqrt{T}})$
CFedAvg [32]	non-convex	$\mathcal{O}(\frac{1}{\sqrt{T}})$	$\|\nabla F(\mathbf{v}_1)\|^2 \leq \mathcal{O}(\sum_{i=1}^{N} \frac{\gamma_i}{(\epsilon_i-\gamma_i)\sqrt{T}} + \frac{1}{T})$
FT-LSGD-DB [33]	non-convex	$\mathcal{O}(\frac{1}{\sqrt{T}})$	$\|\nabla F(\mathbf{v}_2)\|^2 \leq \mathcal{O}(\frac{1}{\sqrt{T}} + \sum_{i=1}^{N} \frac{1}{(1-\gamma_i)^2 T})$

updates of different code lengths are encoded into the same packet. It is necessary to include sufficient information into the header part so that the PS can correctly understand the information contained in a packet. For example, if there are 10 model updates with code length 32, 20 model updates with code length 30, 40 model updates with code length 28. Information of this complicated code method must be contained in the header part, which can significantly expands the size of H. To avoid impairing communication efficacy, we force that all model updates encapsulated into a particular packet r must have the same code length y_r. With this constraint, the header part can be extremely simple since PS only need s and y_r to decode the payload of packet r.

Meanwhile, H is relatively smaller if b_r is bigger. For simplicity, we let $b_r = b$ for all packets where b is the maximum size of each packet. For instance, $b = 1,500$ bytes for a typical TCP packet [34]. Our problem is converted to optimizing the code length for each packet, which implies that the constraint in (2) is changed as

$$P_r(s + y_r) + H \leq b, \quad for\ r = 1, \ldots, R \tag{3}$$

The communication resource is limited, which should be further constrained so that FL will not spend excessive time on the communication of model updates. Let bR denote the budget of uplink communication traffic of a participating client in a communication round. bR is determined by the uplink communication capacity of the client and the total training time budget of FL. Since we focus on the compression algorithm design, bR is simply regarded as a constant in our problem, which has been specified before FL is conducted. Note that the communication cost is also subject to the total number of communication rounds. However, our design can improve existing compression algorithms in FL for every communication round. Thus, the improvement of our design is regardless of the number of conducted communication rounds.

We proceed to formulate the problem to optimize the code length for each packet. In Proposition 1, the constraint of communication packets is not considered. However, the compression error bound will be revised if the code length must be identical for model updates in the same packet. We put k top model updates ranked by a descending order of their magnitudes into R groups, which will be further encapsulated into R packets. The compression error will be specified by:

Proposition 2. *Suppose top k model updates from Client i's $\mathbf{U}_t^i$ at the t-th communication round are selected and encapsulated into R packets. The set of model updates in packet r is denoted by $\mathcal{P}_r$, each of which is encoded with y_r bits. After three compression operations, γ_i indicating the compression error on client i is*

$$\gamma_i = \frac{d^{\beta_i} - (k+1)^{\beta_i}}{d^{\beta_i} - 1} + \frac{1}{d^{\beta_i} - 1} \sum_{r=1}^{R} \Big((\frac{Q(P_r, y_r)}{B^2} + \frac{1}{B_c^2})(Z_r^{\beta_i} - Z_{r-1}^{\beta_i}) \Big), \tag{4}$$

where $k = \sum_{r=1}^{R} P_r$, $Z_r = P_1 + \cdots + P_r$, $Q(P_r, y_r)$ represents the quantization error of model updates in $\mathcal{P}_r$ if each model update is expressed by y_r bits and $B > \max_{\forall r} \frac{Q(P_r,y_r)+1}{2}$. Here, the definitions of β_i and B_c are the same as those in Proposition 1.

The proof of Proposition 2 is similar to that of Proposition 1. Considering that the code length y_r is determined by P_r, *i.e.*, $y_r = \frac{b-H}{P_r} - s$, we use the term $Q(P_r)$ in lieu of $Q(P_r, y_r)$ for simplicity, hereafter.

Our optimization problem is to tune the number of model updates in each packet $P_1, P_2, \ldots, P_R$ to minimize γ_i defined in (4). Formally, we have

$$\begin{aligned} \mathbb{P}1: \min_{P_1,P_2,\ldots,P_R,k} \quad & \gamma_i, \\ s.t. \quad & (3), (4), \sum_{r=1}^{R} P_r = k, 1 \leq k \leq d. \end{aligned} \tag{5}$$

It is non-trivial to solve problem $\mathbb{P}1$ because: 1) the expression of γ_i is very complicated for analyzing its convex property; 2) the search space is too large given $R + 1$ variables, *i.e.*, $P_1, \ldots, P_R, k$.

To solve $\mathbb{P}1$, we add two more constraints to reduce the search space. First, a model update of a larger magnitude should be compressed by a code with more bits, and vice verse. Given that the size of each packet is identical and model updates are encapsulated into R packets by a descending order of their absolute values, it implies that

$$P_1 \leq P_2 \leq \cdots \leq P_R. \tag{6}$$

Second, the search space of k is constrained by a lower bound value $k_{min} = R$, implying single model update in each packet, and a upper bound value $k_{max} = \frac{R(b-H)}{s+1}$, implying that each model update is represented by a single bit. Considering more constraints, our problem is formulated as

$$\mathbb{P}2: \min_{P_1,P_2,\ldots,P_R,k} \quad \gamma_i,$$

$$s.t.\ \ (3),(4),(6),\sum_{r=1}^{R} P_r = k, k_{min} \le k \le k_{max}.$$

Problem $\mathbb{P}2$ can be solved by two steps.

Step 1. Given the range of k, enumerate all values of k.

Step 2. Once k is fixed by step 1, tune $P_1, P_2, \ldots, P_R$ to minimize γ_i.

In step 2, $P_1, P_2, \ldots, P_R$ can be tuned by a series of atomic operations. Before we define the atomic operation, let us consider a special case with only two consecutive packets, *i.e.,* $r-1$ and r, to carry a fixed number of total X model updates. For this special case, we simplify the objective γ_i by discarding all irrelevant constants to get the function $f(X, P_{r-1}, P_r)$, subjecting to $P_{r-1} + P_r = X$, as follows.

$$f(X, P_{r-1}, P_r) = \Big(\frac{Q(P_r)}{B^2} + \frac{1}{B_c^2}\Big)\frac{Z_r^{\beta_i} - Z_{r-1}^{\beta_i}}{d^{\beta_i} - 1} + \Big(\frac{Q(P_{r-1})}{B^2} + \frac{1}{B_c^2}\Big)\frac{Z_{r-1}^{\beta_i} - Z_{r-2}^{\beta_i}}{d^{\beta_i} - 1}. \quad (7)$$

Definition 2. *Atomic Operation: Given $f(X, P_{r-1}, P_r)$, subjecting to $P_{r-1} + P_r = X$, the atomic operation can tune P_r such that $f(X, P_{r-1}, P_r)$ is minimized.*

The atomic operation can be efficiently carried out because of the convexity of $f(X, P_{r-1}, P_r)$.

Lemma 2. *If $Q(P_r)$ is an increasing convex function with respect to P_r, $f(X, P_{r-1}, P_r)$, subjecting to $P_{r-1} + P_r = X$, is a strongly convex function with respect to P_r.*

The proof is presented in Appendix A-C. Although expressions of compression errors are dependent on the specific designs of different quantization algorithms, we can observe their common property that $Q(P_r)$ is an increasing convex function with respect to P_r. Intuitively speaking, if P_r is increased, it implies that more model updates are compressed with a fixed number of bits. Consequently, the compression error is enlarged. Besides, with the increase of P_r, the growth rate of the compression error will become larger and larger. We will use PQ and QSGD, two typical quantization algorithms, to verify this in Appendix A-D.

Theorem 1. *When k is fixed, if there exists $P_1^*, \ldots, P_R^*$ such that $f(X, P_{r-1}^*, P_r^*)$ is minimized for $2 \le r \le R$, $P_1^*, \ldots, P_R^*$ can minimize γ_i in $\mathbb{P}2$ and $P_1^* \le \cdots \le P_R^*$.*

The proof is presented in Appendix A-E. $P_1^*, \ldots, P_R^*$ can be found by Sequential Minimal Optimization (SMO) [35]. SMO was originally designed to optimize a support vector machine (SVM). It breaks an original optimization problem into multiple subproblems by only optimizing two variables each time. After repeatedly selecting and optimizing two variables, it approaches the objective of the SVM. Inspired by SMO, we call atomic operations to optimize the number of model updates in any two consecutive packets for multiple times until all f's get converged implying that we get $P_1^*, \ldots, P_R^*$ for a given k. Finally, the optimal k is selected by the one, denoted by k^*, minimizing γ_i.

Algorithm 1: Optimizing Code Length in Fed-CVLC

1 Input model updates $\mathbf{U}_t^i$ to compute α_i in Definition 1 and initialize $\gamma_i^* \leftarrow \infty$.
2 **for** $k = k_{min}$ ***to*** k_{max} **do**
3 $P_1, \ldots, P_R \leftarrow \frac{k}{R}$.
4 **repeat**
5 **for** $r = 2$ ***to*** R **do**
6 $P_{r-1}, P_r \leftarrow$ Optimize (7).
7 $y_{r-1}, y_r \leftarrow \frac{b-H}{P_{r-1}} - s, \frac{b-H}{P_r} - s$.
8 **end**
9 **until** *Convergence*;
10 Substitute $k, P_1, \ldots, P_R$ into (4) to compute γ_i.
11 **if** $\gamma_i < \gamma_i^*$ **then**
12 $\gamma_i^*, k^*, P_1^*, \ldots, P_R^* \leftarrow \gamma_i, k, P_1, \ldots, P_R$.
13 $y_1^*, \ldots, y_R^* \leftarrow y_1, \ldots, y_R$.
14 **end**
15 **end**
16 **for** $r = 1$ ***to*** R **do**
17 According to the format in Fig. 2, encapsulate P_r^* model updates in $\mathcal{P}_r$, each of which is quantized by y_r^* bits, in the r-th packet.
18 **end**
19 Remaining parameters are set to 0 to get $\hat{\mathbf{U}}_t^i$.

D. Implementation Considerations

We illustrate how to implement our algorithm by encapsulating model updates into packets. Fig. 2 presents the format of packets generated by Fed-CVLC, in which we explain the fields related to Fed-CVLC in a packet as follows.

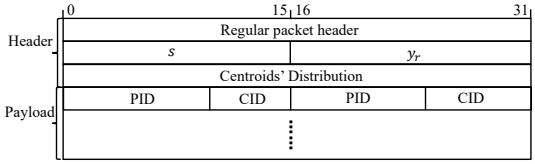

Fig. 2. Packet design when variable-length coding.

1) s and y_r indicate the number of bits to represent position ID and the number of bits for expressing each centriod.

2) Centroid Distribution helps the PS to compute centroids of the quantization algorithm. The centroids are uniformly distributed when using PQ and QSGD for quantization. The distribution of PQ only needs the maximum and minimum values of model updates (consuming 64 bits). QSGD quantizes the absolute values of model updates, which needs to transmit the l2-norm of model updates (consuming 32 bits).

3) PID and CID indicate position ID of model update and centroid ID corresponding to the value of model update.

In our design, the communication overhead is lightweight, which is incurred by the s and y_r fields in the Header part. In Algo. 1, we present the pseudocode to optimize code length in Fed-CVLC. It can be embedded into existing FL algorithms such as FedAvg, conducted per communication round before uploading model updates. In particular, line 6 is the key step to optimize the number of model updates contained in two

consecutive packets. Its time complexity is $\mathcal{O}(\frac{b*R^2}{\log_2 d})$, which is much lower than that of model training. In addition, this workload can be offloaded to the PS if a client with limited capacity transmits the model update distribution to the PS, which can solve $\mathbb{P}2$ and return k^* and P_r^* for compression.

V. Experiment

In this section, we evaluate the performance of Fed-CVLC using public standard datasets and state-of-the-art baselines.

A. Experimental Settings

Datasets. In our experiments, we employ three standard image datasets: CIFAR-10, CIFAR-100 and FEMNIST datasets. For both CIFAR-10 and CIFAR-100, there are 50,000 images as the training set and 10,000 images as the test set. Each image has 3*32*32 pixels. The CIFAR-10 dataset has 10 labels, while the CIFAR-100 dataset has 100 labels. The FEMNIST dataset contains handwritten digital and letter images with 62 labels in total. Each image is with a size of 28*28.

For CIFAR-10 and CIFAR-100, we randomly assign the training set data to clients. Each client will be assigned with 500 samples. By tuning the label assignment, we have both IID and non-IID (not independent and identically distributed) data distributions in our experiments. For the IID data distribution, samples assigned to each client are randomly selected from entire datasets of CIFAR-10 and CIFAR-100, respectively. For the non-IID data distribution, samples assigned to each client are selected from a subset containing only 5 out of 10 labels in CIFAR-10 and 20 out of 100 labels in CIFAR-100. FEMNIST is naturally non-IID since each client only owns images generated by different users. Each client will have 300-400 FEMNIST samples.

Models. The FL task for each image dataset is to train a CNN model for predicting image labels correctly. We train CNN models with 3-layer (number of parameters $3 * 10^5$), 4-layer (number of parameters $5 * 10^6$) and 2-layer (number of parameters $4.5 * 10^5$) to classify CIFAR-10,CIFAR-100 and FEMNIST images, respectively. The structure of each layer is Convolution-BatchNormalization-MaxPooling. We constrain that the number of packets transmitted by each client per communication round is 10 for CIFAR-10 and FEMNIST, and 90 for CIFAR-100. Based on the definition of the maximum transmission unit (MTU) in [34], we set the size of each communication packet to be 1,500 bytes. Our trained CNN models are frequently used by prior works [25], [36].

System Settings. We set up 100 clients and a single PS. In each communication round, 10 out of 100 clients are randomly selected by the PS to participate in training. Each selected client will update models with $E = 5$ local iterations per communication round. We set the learning rate as 0.1 for CIFAR-10, and 0.05 for FEMNIST and CIFAR-100.

Baselines. As we have introduced, combining quantization and sparsification is the state-of-the-art compression technique in FL. We implement such compression algorithms by combining PQ/QSGD [23], [24] and Top_k [27], the well-known sparsification compression algorithm, as our baselines. To be specific, other than the Top_k algorithm, we implement $P(Q)_6Top_k$, $P(Q)_8Top_k$ and $P(Q)_{10}Top_k$ by combining PQ (QSGD) with Top_k where each top model update is quantized with 6, 8 or 10 bits, respectively. We evaluate Fed-CVLC in comparison with baselines from two perspectives: **model accuracy** and **communication traffic**.

B. Experimental Results

Comparing Model Accuracy. In Fig. 3, we compare the model accuracy of each algorithm on different test datasets. In each figure, the x-axis represents the number of conducted communication rounds, while the y-axis represents the model accuracy on the test set. Note that we have fixed the amount of communication traffic (*i.e.* 10 packets per client) for each compression algorithm per communication round so that we can fairly compare the model accuracy of these algorithms. From experiment results presented in Fig. 3, we can draw the following observations:

- Fed-CVLC can steadily achieve the highest model accuracy after every global iteration in all experiment cases under both IID and non-IID data distributions. The improvement of Fed-CVLC indicates the effectiveness to compress model updates with variable-length codes.
- Fed-CVLC is a generic design in that both PQ and QSGD can be applied to outperform corresponding baselines in terms of model accuracy.
- In most cases, combining PQ and Top_k can achieve better model accuracy than that of Top_k, indicating that the design combining quantization and sparsification is the state-of-the-art compression technique, though it is worse than Fed-CVLC.
- The PQ algorithm is slightly better than the QSGD algorithm in terms of model accuracy. In particular, the performance of Q_6Top_k is even worse than that of Top_k because QSGD cannot work well if the number of bits for quantization is too few.

Comparing Communication Traffic. To verify that Fed-CVLC can significantly diminish communication cost, we further compare the total volume of consumed uplink communication traffic by different compression algorithms. In other words, we accumulate the amount of uplink traffic until the trained model can reach the target model accuracy on test sets. The experiment results are presented in Table II. Experiment results shed light on that: 1) Fed-CVLC always consumes the least amount of communication traffic to achieve the target model accuracy in all experiment cases. 2) The last column in table shows the percent of communication traffic further shrunk by Fed-CVLC in comparison with the second best one. It shows that Fed-CVLC can further reduce the communication cost by 16.67%-41.61% based on the state-of-the-art compression technique. 3) Model compression can sheer reduce communication cost. For example, for training the CNN model with 5 million parameters to classify CIFAR-100, the total uplink communication traffic is reduced to 128.75MB over the entire training process.

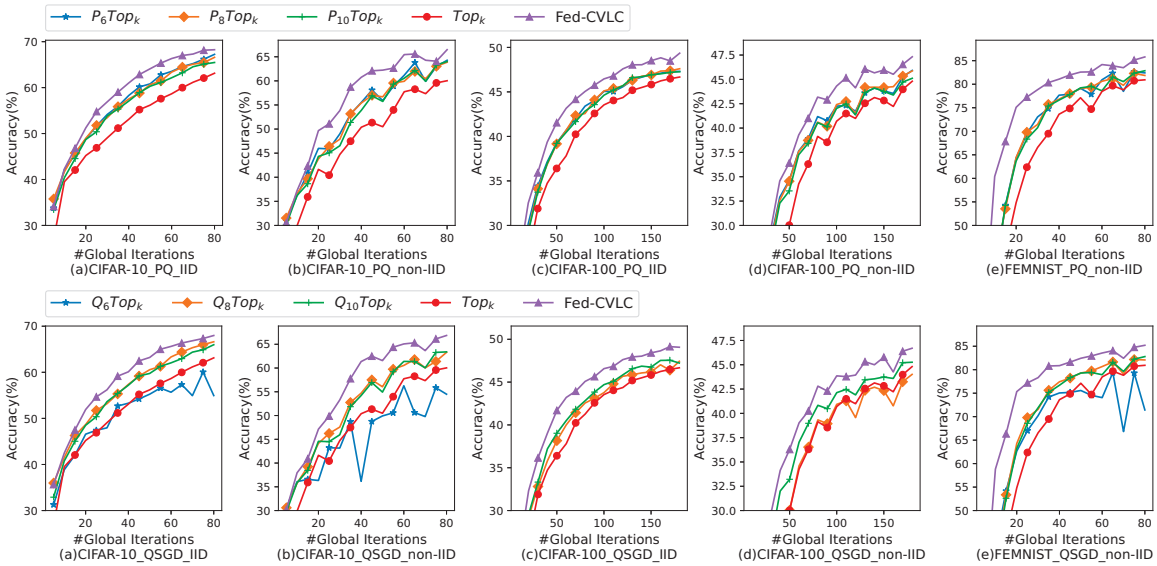

Fig. 3. Comparison of model accuracy on the test set of CIFAR-10, CIFAR-100 and FEMNIST when using PQ(Top) and QSGD(Bottom) for quantization.

TABLE II
COMPARISON OF TOTAL UPLINK COMMUNICATION TRAFFIC (MB) CONSUMED BY DIFFERENT COMPRESSION ALGORITHMS TO REACH TARGET MODEL ACCURACY WHEN USING PQ(TOP) AND QSGD(BOTTOM) FOR QUANTIZATION.

	Fed-CVLC	Top_k	P_6Top_k	P_8Top_k	$P_{10}Top_k$	Reduced % by ours
CIFAR-10+IID (63%)	**7.15**	11.44	8.58	8.58	9.30	16.67%
CIFAR-10+non-IID (60%)	**5.72**	11.44	8.58	9.30	8.58	33.33%
FEMNIST (80%)	**5.01**	10.73	8.58	8.58	9.30	41.61%
CIFAR-100+IID (46%)	**128.75**	205.99	167.37	167.37	167.37	23.07%
CIFAR-100+non-IID (44%)	**128.75**	231.74	180.24	167.37	180.24	23.07%

	Fed-CVLC	Top_k	Q_8Top_k	$Q_{10}Top_k$	$Q_{12}Top_k$	Reduced % by ours
CIFAR-10+IID(63%)	**7.15**	11.44	8.58	10.01	9.30	16.67%
CIFAR-10+non-IID(60%)	**5.72**	11.44	8.58	8.58	8.58	33.33%
FEMNIST (80%)	**5.01**	10.73	8.58	9.30	8.58	41.61%
CIFAR-100+IID(46%)	**128.75**	205.99	180.24	167.37	167.37	23.07%
CIFAR-100+non-IID(44%)	**167.37**	231.74	231.74	218.87	218.87	23.53%

VI. CONCLUSION

FL is a cutting-edge technique in preserving data privacy during the training of machine learning models. Different from traditional machine learning, FL will not relocate data samples, which however can incur heavy communication overhead. Until now, quantization and sparsification are two most popular compression approaches to prohibiting communication cost of FL. Yet, existing compression algorithms rigidly set a fixed code length for model compression, restricting the effectiveness of compression. In this work, we made an initial attempt to expand the compression design space in FL by generalizing quantization and sparsification with variable-length codes. Not only we considered practical implementation of communication packets in our design, but also conducted rigorous analysis to guarantee the performance of our algorithm. Comprehensive experiments have been conducted with public datasets to demonstrate that Fed-CVLC can considerably shrink clients' uplink communication traffic in FL. In comparison with the state-of-the-art technique, Fed-CVLC can further shrink communication traffic by 27.64% or improve model accuracy by 3.21% on average. Our future work is to extend Fed-CVLC to make it applicable in more advanced FL systems such as decentralized FL or hierarchical FL systems.

APPENDIX A

A. Proof of Lemma 1

We assume that a vector $\overline{\mathbf{U}}$ has z_y elements, which are quantized by y bits, to get vector $\widetilde{\mathbf{U}}$. Each element in vector $\mathbf{U}$ is further scaled by dividing a factor B to obtain vector $\hat{\mathbf{U}}$. The error between $\overline{\mathbf{U}}$ and $\hat{\mathbf{U}}$ is therefore bounded by:

$$\begin{aligned}\mathbb{E}\|\hat{\mathbf{U}}-\overline{\mathbf{U}}\|^2 &= \mathbb{E}\|\frac{\widetilde{\mathbf{U}}}{B}-\overline{\mathbf{U}}\|^2 \\ &\overset{(a)}{=} \mathbb{E}\|\frac{\widetilde{\mathbf{U}}-\overline{\mathbf{U}}}{B}\|^2+\mathbb{E}\|(1-\frac{1}{B})\overline{\mathbf{U}}\|^2 \\ &\overset{(b)}{\leq} \frac{Q(z_y,y)+(B-1)^2}{B^2}\|\overline{\mathbf{U}}\|^2, \end{aligned} \tag{8}$$

where equality (a) holds because the quantization algorithm is unbiased. Inequality (b) holds because $Q(z_y, y)$ represents the error by quantizing each element with y bits for total z_y elements, *i.e.*,$\mathbb{E}\|\widetilde{\mathbf{U}}-\overline{\mathbf{U}}\|^2 \leq Q(z_y,y)\|\overline{\mathbf{U}}\|^2$.

B. Proof of Proposition 1

We start from analyzing the compression error caused by variable-length coding. The error $\|\hat{\mathbf{U}}_t^i-\mathbf{U}_t^i\|^2$ has two components: sparsification error and quantization error with

variable-length codes for Y different groups. By substituting the power law distribution in Definition 1 into $U_t^i\{l\}$, we have

$$\mathbb{E}\frac{\|\hat{\mathbf{U}}_t^i-\mathbf{U}_t^i\|^2}{\|\mathbf{U}_t^i\|^2}=\mathbb{E}\frac{\sum_{y=1}^{Y}\|\frac{\widetilde{\mathbf{U}}_{ty}^i}{B}-\mathbf{U}_{ty}^i\|^2+\sum_{l=k+1}^{d}(U_t^i\{l\})^2}{\|\mathbf{U}_t^i\|^2}$$
$$\leq\frac{\sum_{y=1}^{Y}\frac{Q(z_y,y)+(B-1)^2}{B^2}\|\mathbf{U}_{ty}^i\|^2}{\|\mathbf{U}_t^i\|^2}+\frac{\sum_{l=k+1}^{d}(U_t^i\{l\})^2}{\|\mathbf{U}_t^i\|^2}$$
$$\approx\frac{d^{2\alpha_i+1}-(k+1)^{2\alpha_i+1}}{d^{2\alpha_i+1}-1}+\frac{1}{d^{2\alpha_i+1}-1}\sum_{y=1}^{Y}$$
$$\big((\frac{Q(z_y,y)}{B^2}+\frac{1}{B_c^2})(Z_y^{2\alpha_i+1}-Z_{y+1}^{2\alpha_i+1})\big),\tag{9}$$

where $Z_{y+1}=z_Y+z_{Y-1}+\cdots+z_{y+1}$ and $\mathbf{U}_{ty}^i$ is a z_y-dimension vector. Each element in $\mathbf{U}_{ty}^i$ will be quantized with y bits. The elements in the vector are $U_t^i\{l\}'s$ where $l=Z_{y+1}+1,Z_{y+1}+2,\ldots,Z_{y+1}+z_y$. To ensure that the trained model converges, we need to make the compression algorithm satisfy the condition that $\mathbb{E}\frac{\|\hat{\mathbf{U}}_t^i-\mathbf{U}_t^i\|^2}{\|\mathbf{U}_t^i\|^2}\leq\gamma_i$,where $0<\gamma_i<1$. This conditions implies that the compression is bounded such that the trained model will not diverge in the end. Therefore, we have $B>\max_{\forall y}\frac{Q(z_y,y)+1}{2}$ to guarantee $0<\gamma_i<1$.

C. Proof of Lemma 2

Recall that $Z_r=P_1+\cdots+P_r$ and $Z_{r-2}=P_1+\cdots+P_{r-2}$. X is a fixed number, and thus $P_r+P_{r-1}=X$ is fixed. We can get $f(P_r)=(\frac{Q(P_r)}{B^2}+\frac{1}{B_c^2})\frac{Z_r^{\beta_i}-(Z_r-P_r)^{\beta_i}}{d^{\beta_i}-1}+(\frac{Q(X-P_r)}{B^2}+\frac{1}{B_c^2})\frac{(Z_r-P_r)^{\beta_i}-Z_{r-2}^{\beta_i}}{d^{\beta_i}-1}$. We derive the above equation to have:

$$f'(P_r)=\underbrace{\frac{Q'(P_y)}{B^2}*\frac{Z_r^{\beta_i}-(Z_r-P_r)^{\beta_i}}{d^{\beta_i}-1}}_{A_1}$$
$$+\underbrace{(\frac{Q(P_r)}{B^2}+\frac{1}{B_c^2})\frac{\beta_i(Z_r-P_r)^{\beta_i-1}}{d^{\beta_i}-1}}_{A_2}$$
$$-\underbrace{\frac{Q'(X-P_r)}{B^2}*\frac{(Z_r-P_r)^{\beta_i}-Z_{r-2}^{\beta_i}}{d^{\beta_i}-1}}_{A_3}$$
$$-\underbrace{(\frac{Q(X-P_r)}{B^2}+\frac{1}{B_c^2})\frac{\beta_i(Z_r-P_r)^{\beta_i-1}}{d^{\beta_i}-1}}_{A_4}.\tag{10}$$

Here $Q'(P_r)$ is the derivative of the quantization error. Since $Q(P_y)$ is an increasing strongly convex function, $Q'(P_y)$ is monotonically increasing. Moreover, the derivative of $\frac{Z_r^{\beta_i}-(Z_r-P_r)^{\beta_i}}{d^{\beta_i}-1}$ is a monotonically increasing function. Since the two terms multiplied in A_1 are greater than 0, A_1 is monotonically increasing. Similarly, we get that A_3 is monotonically decreasing. Therefore A_1-A_3 is monotonically increasing. Similar to the analysis of A_1-A_3, we have that A_2-A_4 is monotonically increasing. Since A_1-A_3 and A_2-A_4 are both monotonically increasing, it manifests that $f''(P_r)>0$, and hence $f(P_r)$ is a strongly convex function.

We proceed to discuss why $P_r>P_{r-1}$. If $P_r<\frac{X}{2}$, it means $P_r<P_{r-1}$ and $P_r<X-P_r$. There are $\frac{Q'(P_y)}{B^2}<\frac{Q'(X-P_r)}{B^2}$ and $\frac{Z_r^{\beta_i}-(Z_r-P_r)^{\beta_i}}{d^{\beta_i}-1}<\frac{(Z_r-P_r)^{\beta_i}-Z_{r-2}^{\beta_i}}{d^{\beta_i}-1}$ in this scenario. Hence, we have $A_1-A_3<0$ and $A_2-A_4<0$, suggesting that $f(P_r)$ is a decreasing function if $P_r<\frac{X}{2}$, and thus $f(P_r)$ can be decreased if we increase P_r until $P_r>\frac{X}{2}$. Thus, the optimal P_r must be in the range $(\frac{X}{2},X)$, implying that $P_r>P_{r-1}$.

D. Case Study

We show that quantization error $Q(P_r)$ is an increasing convex function by conducting a case study with two typical quantization algorithms, namely PQ and QSGD algorithms.

In the PQ [23] algorithm, centroids are uniformly distributed within the value range of model updates. Each model update within each interval is mapped to its upper or lower centroids. The quantization error of PQ is $Q(P_r)=\frac{P_r}{(2^{\frac{b-H}{P_r}-s}-1)^2}$ [23].

Theorem 2. *The quantization error $Q(P_r)$ of PQ is an increasing convex function with respect to P_r.*

Proof. We can obtain $Q'(P_r)=\frac{2^{\frac{b-H}{P_r}-s+1}*\ln 2*(b-H)}{(2^{\frac{b-H}{P_r}-s}-1)^3P_r}+\frac{1}{(2^{\frac{b-H}{P_r}-s}-1)^2}>0$. The second term in $Q'(P_r)$ is increasing as P_r increases, so we analyze the first term separately. The numerator of the derivative of the first term is $3[2^{\frac{b-H}{P_r}-s}-1]^2*2^{\frac{b-H}{P_r}-s}*\ln 2*\frac{b-H}{P_r}*2^{\frac{b-H}{P_r}-s+1}-2^{\frac{b-H}{P_r}-s+1}[2^{\frac{b-H}{P_r}-s}-1]^3*[\ln 2*\frac{b-H}{P_r}+1]>0$. This proves that $Q''(P_r)>0$ and shows that $Q(P_r)$ is an increasing convex function. □

QSGD is another typical quantization algorithm. In [24], its quantization error is $Q(P_r)=\min(Q_1(P_r),Q_2(P_r))$, where $Q_1(P_r)=\frac{P_r}{2^{2(\frac{b-H}{P_r}-s)}}$ and $Q_2(P_r)=\frac{\sqrt{P_r}}{2^{\frac{b-H}{P_r}-s}}$.

Theorem 3. *Both $Q_1(P_r)$ and $Q_2(P_r)$ are increasing convex functions with respect to P_r.*

The proof is similar to that of Theorem 2, and thus omitted.

E. Proof of Theorem 1

We will prove Theorem 1 by a contradiction. Considering that $f(X,P_{r-1},P_r)$ is a strongly convex function, there is only one optimal solution to minimize $f(X,P_{r-1},P_r)$ for $\forall r$. Thus, the sequence $P_1^*,\ldots,P_R^*$ that minimizes $f(X,P_{r-1},P_r)$ for $2\leq r\leq R$ is unique. We use γ_i^* to denote the compression error in this case. If there exists another sequence $P_1',\ldots,P_R'$ and its corresponding error is $\gamma_i'<\gamma_i^*$. There must be a pair of P_{r-1}' and P_r' in this sequence that cannot minimize the function $f(X,P_{r-1},P_r)$. Therefore, the number of model updates encapsulated into these two packets can be adjusted to obtain P_{r-1}'' and P_r'' so that $f(X,P_{r-1},P_r)$ is further reduced to achieve a lower error $\gamma_i''<\gamma_i'$. The sequence of γ_i'' continues to adjust the assignment of each two adjacent packets to gradually reduce the error, which will eventually result in $\gamma_i''<\gamma_i'$, contradicting the assumption that γ_i' is the minimum value.

Thus, $P_1^*,\ldots,P_R^*$ that minimize the f function for any two adjacent packets can achieve the minimum error γ_i^*.

REFERENCES

[1] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–19, 2019.

[2] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics (AISTATS)*, 2017, pp. 1273–1282.

[3] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, and C. Miao, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Communications Surveys & Tutorials (COMST)*, vol. 22, no. 3, pp. 2031–2063, 2020.

[4] P. Wang, F. Ye, and X. Chen, "A Smart Home Gateway Platform for Data Collection and Awareness," *IEEE Communications Magazine (MCOM)*, vol. 56, no. 9, pp. 87–93, 2018.

[5] J. Hamer, M. Mohri, and A. T. Suresh, "Fedboost: A communication-efficient algorithm for federated learning," in *International Conference on Machine Learning (ICML)*, 2020, pp. 3973–3983.

[6] D. Rothchild, A. Panda, E. Ullah, N. Ivkin, I. Stoica, V. Braverman, J. Gonzalez, and R. Arora, "Fetchsgd: Communication-efficient federated learning with sketching," in *International Conference on Machine Learning (ICML)*, 2020, pp. 8253–8265.

[7] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," in *Annual Conference on Neural Information Processing Systems (NeurIPS) Workshop on Private Multi-Party Machine Learning*, 2016, pp. 1–5.

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE conference on computer vision and pattern recognition (CVPR)*, 2016, pp. 770–778.

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is All you Need," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017, pp. 1–11.

[10] W. Wen, C. Xu, F. Yan, C. Wu, Y. Wang, Y. Chen, and H. Li, "TernGrad: Ternary Gradients to Reduce Communication in Distributed Deep Learning," in *Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2017, pp. 1509–1519.

[11] J. Bernstein, Y.-X. Wang, K. Azizzadenesheli, and A. Anandkumar, "signSGD: Compressed optimisation for non-convex problems," in *International Conference on Machine Learning (ICML)*, 2018, pp. 560–569.

[12] Y. Lin, S. Han, H. Mao, Y. Wang, and B. Dally, "Deep Gradient Compression: Reducing the Communication Bandwidth for Distributed Training," in *International Conference on Learning Representations (ICLR)*, 2018, pp. 1–14.

[13] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Robust and Communication-Efficient Federated Learning From Non-i.i.d. Data," *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)*, vol. 31, no. 9, pp. 3400–3413, 2020.

[14] A. M Abdelmoniem, A. Elzanaty, M.-S. Alouini, and M. Canini, "An efficient statistical-based gradient compression technique for distributed training systems," *Machine Learning and Systems (MLSys)*, vol. 3, pp. 297–322, 2021.

[15] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the Convergence of FedAvg on Non-IID Data," in *International Conference on Learning Representations (ICLR)*, 2020, pp. 1–26.

[16] H. Yang, M. Fang, and J. Liu, "Achieving Linear Speedup with Partial Worker Participation in Non-IID Federated Learning," in *International Conference on Learning Representations (ICLR)*, 2021, pp. 1–23.

[17] S. Chen and B. Li, "Towards Optimal Multi-Modal Federated Learning on Non-IID Data with Hierarchical Gradient Blending," in *IEEE Conference on Computer Communications (INFOCOM)*, 2022, pp. 1469–1478.

[18] Z. Wang, Y. Zhu, D. Wang, and Z. Han, "Fedfpm: A unified federated analytics framework for collaborative frequent pattern mining," in *IEEE Conference on Computer Communications (INFOCOM)*, 2022, pp. 61–70.

[19] Y. Liu, L. Xu, X. Yuan, C. Wang, and B. Li, "The right to be forgotten in federated learning: An efficient realization with rapid retraining," in *IEEE Conference on Computer Communications (INFOCOM)*, 2022, pp. 1749–1758.

[20] J. Perazzone, S. Wang, M. Ji, and K. S. Chan, "Communication-Efficient Device Scheduling for Federated Learning Using Stochastic Optimization," in *IEEE Conference on Computer Communications (INFOCOM)*, 2022, pp. 1449–1458.

[21] B. Luo, W. Xiao, S. Wang, J. Huang, and L. Tassiulas, "Tackling System and Statistical Heterogeneity for Federated Learning with Adaptive Client Sampling," in *IEEE Conference on Computer Communications (INFOCOM)*, 2022, pp. 1739–1748.

[22] Z. Wang, H. Xu, J. Liu, H. Huang, C. Qiao, and Y. Zhao, "Resource-Efficient Federated Learning with Hierarchical Aggregation in Edge Computing," in *IEEE Conference on Computer Communications (INFOCOM)*, 2021, pp. 1–10.

[23] A. T. Suresh, X. Y. Felix, S. Kumar, and H. B. McMahan, "Distributed mean estimation with limited communication," in *International Conference on Machine Learning (ICML)*, 2017, pp. 3329–3337.

[24] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "QSGD: Communication-efficient SGD via gradient quantization and encoding," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017, pp. 1709–1720.

[25] R. Hönig, Y. Zhao, and R. Mullins, "DAdaQuant: Doubly-adaptive quantization for communication-efficient Federated Learning," in *International Conference on Machine Learning (ICML)*, 2022, pp. 8852–8866.

[26] J. Wangni, J. Wang, J. Liu, and T. Zhang, "Gradient sparsification for communication-efficient distributed optimization," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2018, pp. 1299–1309.

[27] S. U. Stich, J.-B. Cordonnier, and M. Jaggi, "Sparsified SGD with memory," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2018, pp. 4447–4458.

[28] X. Qian, P. Richtárik, and T. Zhang, "Error compensated distributed SGD can be accelerated," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 34, pp. 30 401–30 413, 2021.

[29] D. Basu, D. Data, C. Karakus, and S. Diggavi, "Qsparse-local-SGD: Distributed SGD with Quantization, Sparsification and Local Computations," in *Advances in Neural Information Processing Systems (NeurIPS)*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32, 2019, pp. 1–12.

[30] 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 (JSAC)*, vol. 37, no. 6, pp. 1205–1221, 2019.

[31] B. Luo, X. Li, S. Wang, J. Huang, and L. Tassiulas, "Cost-Effective Federated Learning Design," in *International Conference on Computer Communications (INFOCOM)*. IEEE, 2021, pp. 1–10.

[32] H. Yang, J. Liu, and E. S. Bentley, "CFedAvg: Achieving Efficient Communication and Fast Convergence in Non-IID Federated Learning," in *International Symposium on Modeling and Optimization in Mobile, Ad hoc, and Wireless Networks (WiOpt)*, 2021, pp. 1–8.

[33] L. Li, D. Shi, R. Hou, H. Li, M. Pan, and Z. Han, "To Talk or to Work: Flexible Communication Compression for Energy Efficient Federated Learning over Heterogeneous Mobile Edge Devices," in *IEEE International Conference on Computer Communications (INFOCOM)*, 2021, pp. 1–10.

[34] N. Islam, C. C. Bawn, J. Hasan, A. I. Swapna, and M. S. Rahman, "Quality of service analysis of Ethernet network based on packet size," *Journal of Computer and communications (JCC)*, vol. 4, no. 4, pp. 63–72, 2016.

[35] J. Platt, "Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines," Microsoft, Tech. Rep. MSR-TR-98-14, April 1998.

[36] H. Wang, Z. Kaplan, D. Niu, and B. Li, "Optimizing Federated Learning on Non-IID Data with Reinforcement Learning," in *International Conference on Computer Communications (INFOCOM)*. IEEE, 2020, pp. 1698–1707.