Achieving Lossless Gradient Sparsification via Mapping to Alternative Space in Federated Learning

Do-Yeon Kim [1] **Dong-Jun Han** [2] **Jun Seo** [3] **Jaekyun Moon** [1]

Abstract

Handling the substantial communication burden in federated learning (FL) still remains a significant challenge. Although recent studies have attempted to compress the local gradients to address this issue, they typically perform compression only within the original parameter space, which may potentially limit the fundamental compression rate of the gradient. In this paper, instead of restricting our scope to a fixed traditional space, we consider an alternative space that provides an improved compressibility of the gradient. To this end, we utilize the structures of input activation and output gradient in designing our mapping function to a new space, which enables *lossless gradient sparsification*, i.e., mapping the gradient to our new space induces a greater number of *near-zero* elements without any loss of information. In light of this attribute, employing sparsification-based compressors in our new space allows for more aggressive compression with minimal information loss than the baselines. More surprisingly, our model even reaches higher accuracies than the full gradient uploading strategy in some cases, an extra benefit for utilizing the new space. We also theoretically confirm that our approach does not alter the existing, best known convergence rate of FL thanks to the orthogonal transformation properties of our mapping.

1. Introduction

Federated learning (FL) is a well-established approach based on which a large number of clients collaborate to construct a global prediction model while keeping their local data private (McMahan et al., 2017; Bonawitz et al., 2019; Li et al., 2020a; Zhu et al., 2021). Due to its effectiveness, FL has been extensively deployed in many privacy-sensitive applications, such as healthcare system, financial services, and autonomous driving (Antunes et al., 2022; Li et al., 2021b; Aurna et al., 2023; Nguyen et al., 2022; Zhang et al., 2021). During FL training, the participating nodes repeatedly communicate their weight updates with the server instead of directly transmitting their individual data. However, uploading the gradients to the server every round can potentially impose a significant communication burden, particularly on resource-constrained network edges, such as mobile phones, drones, and Internet of Things (IoT) devices.

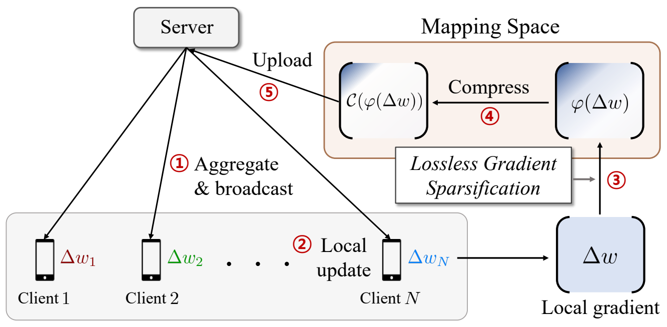
To reduce the communication load during the uplink transmission, previous works have attempted to compress the local updates based on various types of approaches including quantization, sparsification, low-rank compression, to name a few (Li et al., 2021a; Hyeon-Woo et al., 2022; Rothchild et al., 2020). Quantization-based methods compress the gradients by representing them with low bit-width (e.g. $1 \sim 4$ bits), resulting in a smaller communication burden compared to full precision of floating point (Alistarh et al., 2017; Reisizadeh et al., 2020; Jhunjhunwala et al., 2021). Sparsification, or the Top-k algorithm, selects the largest k elements based on their magnitudes and let the remaining ones zero (Wangni et al., 2018; Shi et al., 2019). Some works employed both Top-k and quantization (Sattler et al., 2019; Li & Li, 2023) in pursuit of more aggressive compression during each communication round.

Motivation. Despite the numerous prior efforts that have been made on minimizing communication overheads, majority of such endeavors have tended to consider only the original parameter space. Specifically, most existing methods compressed the weight updates *only within* the space where the gradient vector *originally lies in*. However, limiting the scope to a fixed, certain space may potentially restrict the fundamental compression rate of the gradient. In this paper, we turn our focus on exploring an alternative space in which we can endow the gradient with higher compressibility. In short, our goal in this work boils down to answering the following questions:

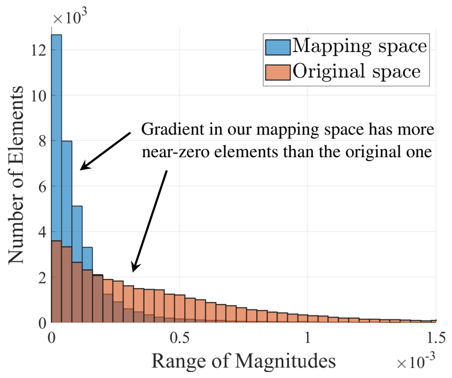
Is there any alternative space suitable for more aggressive yet reliable compression beyond the original space? If

[1]Korea Advanced Institute of Science and Technology (KAIST) [2]Purdue University [3]LG AI Research. Correspondence to: Dong-Jun Han <han762@purdue.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).



(a) Overview of our approach with mapping space



(b) Histogram of gradient magnitudes

Figure 1: **(a)** An overview of our approach in FL setup. The numbers in red circles indicate the order of the entire process. As our mapping space is designed to enable lossless gradient sparsification, we can compress the gradient, which is denoted by Δw, without significant information loss in the subsequent compression. **(b)** Histogram of the gradient magnitudes in both original and mapping space for a specific layer of initialized ResNet18 on CIFAR100. Each bar represents the number of elements within a certain binwidth range.

so, can we find a mapping function that can transfer the gradient to such a new space?

Goal & Overview. In this paper, we aim to construct a new mapping function, designed to be exploited by individual clients (as in the third step of Figure 1a) to achieve lossless gradient sparsification – sparsification in a sense that most elements of the gradient become near-zero without any loss of information. To showcase the effectiveness of the proposed solution, we visualize the distributions of gradient magnitudes at a specific layer in both the original space and the alternative space constructed by our mapping function. As shown in Figure 1b, the gradient in the mapping space tends to have more near-zero elements than its original counterpart. Hence, the information of the gradient can be mostly captured with a remaining, relatively fewer number of elements in the mapping space. This property is a significant advantage of the proposed approach since the near-zero elements can be removed in the alternative space with negligible performance loss. As a result, applying existing sparsification-based compressors (e.g., Top-k, Sparse-Binary) in this new space facilitates more effective compression compared to applying them in the original space, enhancing communication efficiency in FL.

Main Contribution. To design a mapping space that achieves the above property, we start by rewriting the weight gradient of a specific layer as a multiplicative form of two vectors: (i) the input activation and (ii) the gradient with respect to the output of the layer (hereafter referred to as the "output gradient" throughout the paper). We then manipulate these vectors towards having a greater sparsity. Specifically, we capture the leading components for both input activation and the output gradient via singular value decomposition (SVD), and then make majority of information contained in these vectors to be aligned with them. This makes the gradient sparser than the original one as shown in the histograms in Figure 1b, allowing for more aggressive compression in the alternative space. In addition, we can formalize our mapping to be an orthogonal transformation, thereby ensuring that the amount of information is not compromised during the mapping process. We also theoretically verify that our approach does not alter the existing convergence rate of FL as we design our mapping to inherit all the desired properties of the orthogonal transformation.

Throughout extensive simulations, we demonstrate that our approach effectively minimizes the uplink communication load required to reach the desired accuracy compared to the baselines. In particular, our proposed mapping space enhances the communication efficiency of the existing sparsification-based compressors thanks to the lossless gradient sparsification property. Even more surprisingly, our model achieves a final accuracy higher than the conventional FedAvg model without compression, especially when Sparse-Binary compressor is employed, which is not the case when adopting it in the original space.

2. Preliminaries

2.1. Federated Learning

Let us define the local dataset of each client $n = 1, 2, \dots N$ as D_n where N is the total number of clients collaborating in the federated learning (FL) system. We also let $F_n(w) := \frac{1}{|D_n|}\sum_{x \in D_n} \ell(x; w)$ be the loss function computed at each n-th client using D_n, where $w \in \mathbb{R}^d$ denotes the parameter of the model. The goal of FL is to train a global prediction model, aiming to solve the following optimization problem of the form (McMahan et al., 2017; Li et al., 2020b): $\min_{w \in \mathbb{R}^d} F(w) = \frac{1}{N}\sum_{n=1}^{N} F_n(w)$.

During FL, each n-th client updates the model and sends its local update Δw_n to central server. The server then aggregates all the received gradients as $\Delta w = \frac{1}{N}\sum_{n=1}^{N} \Delta w_n$

and broadcasts Δw to all the participating clients. After receiving Δw sent from the server, each client updates the model with $w \leftarrow w + \Delta w$, synchronizing it to global model across all the clients. This process repeats until the global model converges. Note that in the case of conducting multiple local updates, the gradient is obtained from difference between the parameter before and after the local training, i.e., $\Delta w_n = w_n - w$, where w_n denotes the parameter obtained from multiple local updates starting from w.

2.2. Gradient Compression

As aforementioned, a sparsification-based compression appears to be highly compatible with our mapping approach due to its sparsifying property. Here we consider the Top-k (Stich et al., 2018) and Sparse-Binary (Sattler et al., 2019; Li & Li, 2023) compressors, both of which are well-recognized sparsification-based schemes in the FL literature.

Definition 2.1 (Top-k)**.** For any vector $g \in \mathbb{R}^d$, Top-k operator $\mathcal{C}^{\mathrm{k}}(\cdot)$ is defined as

$$[\mathcal{C}^{\mathrm{k}}(g)]_i \leftarrow \begin{cases} [g]_i & \text{if } [|g|]_i \geq v_k \\ 0 & \text{otherwise} \end{cases}, \quad \forall i \in \{1, 2, \ldots, d\},$$

where $[\cdot]_i$ represents the i-th element of the vector, and v_k denotes the k-th largest value among all the elements of $|g|$.

Definition 2.2 (k-Sparse-Binary)**.** For any vector $g \in \mathbb{R}^d$, k-Sparse-Binary operator $\mathcal{C}^{k\mathrm{SB}}(\cdot)$ is defined as

$$[\mathcal{C}^{k\mathrm{SB}}(g)]_i \leftarrow \begin{cases} \frac{\|\mathcal{C}^{\mathrm{k}}(g)\|_1}{\|\mathcal{C}^{\mathrm{k}}(g)\|_0} \cdot \mathrm{Sign}([\mathcal{C}^{\mathrm{k}}(g)]_i) & \text{if } [\mathcal{C}^{\mathrm{k}}(g)]_i \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

for all $i \in \{1, 2, \ldots, d\}$, where $\| \cdot \|_0$ denotes the number of non-zero elements and Sign$(\cdot)$ refers to the sign operator.

In words, the Top-k compressor selects the k (which is much smaller than d) elements based on their magnitudes and sets the remaining ones to zero. The Sparse-Binary scheme is designed for a more aggressive compression beyond the Top-k method that quantizes non-zero elements to a 1 bit-width right after the Top-k operation. In our simulations regarding $\mathcal{C}^{k\mathrm{SB}}(\cdot)$, the magnitude $\frac{\|\mathcal{C}^{\mathrm{k}}(g)\|_1}{\|\mathcal{C}^{\mathrm{k}}(g)\|_0}$ is computed in the layer-wise manner as suggested in (Zheng et al., 2019; Li & Li, 2023).

We also quantify the amount of error induced by the gradient compression (also known as compression error) by means of its norm, which has been adopted as a common practice in many existing works (Reisizadeh et al., 2020; Qu et al., 2022a; Li & Li, 2023; Horváth & Richtarik, 2021).

Definition 2.3. Given any compressor $\mathcal{C}(\cdot)$, the compressor is said to induce a compression error to the extent of at most $1 - \rho$ with respect to its l_2 norm, if $\mathbb{E}[\|g - \mathcal{C}(g)\|_2^2 | g] \leq (1 - \rho)\|g\|_2^2$ holds[1] for all $g \in \mathbb{R}^d$ where $0 < \rho \leq 1$.

For Top-k as an example, it is obvious that selecting smaller k elements results in a smaller ρ, implying a greater information loss. In a later discussion, we will demonstrate that the trade-off between k and ρ in our mapping space is considerably milder than in the original space.

3. Proposed Method

In this section, we provide a detailed description of our approach. First, we explain how our mapping operation works towards lossless gradient sparsification, and then outline the overall procedure for utilizing the mapping in communication-efficient FL setting.

3.1. Lossless Gradient Sparsification

We first start by revisiting how the gradient of[2] weight matrix is derived for a specific layer. Let W_l and $\mathbf{a}_l$ be the weight matrix and the input activation for the l-th layer, respectively. Then the forward propagation is given by $\mathbf{z}_l = W_l \mathbf{a}_l$, where $\mathbf{z}_l$ denotes the output of the l-th layer. By the chain rule, the gradient of W_l can be derived as follows:

$$dW_l = d\mathbf{z}_l \cdot \mathbf{a}_l^\top, \tag{1}$$

where dW_l and $d\mathbf{z}_l$ represent the derivative of loss *w.r.t.* W_l and $\mathbf{z}_l$ respectively. In other words, the weight gradient can be expressed in terms of both the input activation and output gradient as the form of their outer product. Thus, if we can align the majority of information with only a few number of elements for these vectors, i.e., most elements become near-zero, the weight gradient may comprise a larger number of near-zero as well. Drawing on the insight, we consider the singular value decomposition (SVD) of the covariance matrices of activations and the output gradients as follows:

$$A_l A_l^\top = P_l \Lambda_l P_l^\top \quad \text{and} \quad Z_l Z_l^\top = Q_l D_l Q_l^\top, \tag{2}$$

where A_l and Z_l refer to the matrices constructed by stacking $\mathbf{a}_l$ and $d\mathbf{z}_l$ regarding multiple data points in column-wise manner respectively, and Λ_l, D_l are the diagonal matrices having singular values, and P_l, Q_l are the corresponding orthogonal matrices. Now we define our mapping operation $\varphi_l(\cdot)$ as the multiplications of P_l and Q_l on both sides of the gradient as shown below:

$$\varphi_l(dW_l) := Q_l^\top dW_l P_l = Q_l^\top d\mathbf{z}_l \cdot \mathbf{a}_l^\top P_l. \tag{3}$$

Note that both P_l and Q_l consist of leading principal components, constituting certain subspaces in which most of the input activations and the output gradients are concentrated.

[1]Note that the expectation $\mathbb{E}[\cdot]$ is taken over the compressor $\mathcal{C}(\cdot)$. We can omit this expectation if the compressor is deterministic, e.g., Top-k and k-Sparse-Binary.

[2]We will interchangeably use "gradient of" to refer to "gradient with respect to (*w.r.t.*)" for brevity in the paper.

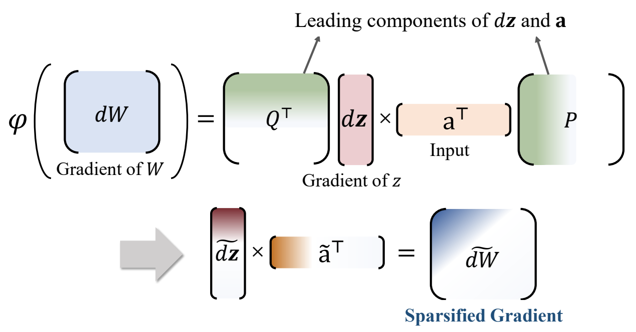


Figure 2: An illustrative figure of how our mapping function works. The color intensity indicates the magnitude of each element. Since both P and Q contain the leading components for $\mathbf{a}$ and dz respectively, most of the elements in $\tilde{\mathbf{a}}$ and $d\tilde{\mathbf{z}}$ are aligned with the fewer components than the original counterpart.

Multiplying P_l to $\mathbf{a}_l$ make the values in vector $\mathbf{a}_l$ focused on certain elements, and hence $\mathbf{a}_l^\top P_l$ becomes more sparse than $\mathbf{a}_l$. This similarly applies to $d\mathbf{z}_l$ and Q_l. As a result, their outer product, i.e., the gradient of the weight, would be more sparsified in the mapping space than in the original one. Figure 2 visually shows how our proposed mapping leads to a sparse gradient. The vector with a tilde on top of it denotes itself in the mapping space after the transformation.

Since P_l and Q_l are orthogonal matrices, our mapping operation does not compromise the amount of information at all, i.e., preserves the l_2 distance. In other words, our mapping admits its inverse function, denoted by $\varphi_l^{-1}(\cdot)$, which allows us to fully recover the original one as follows:

$$\varphi_l^{-1}(\varphi_l(dW_l)) := Q_l \varphi_l(dW_l) P_l^\top = dW_l. \tag{4}$$

Eq. (4) implies that the information loss in the gradient only occurs during the subsequent compression, while full information is retained during mapping or inverse mapping process. Note that once we prepare P_l and Q_l for forward mapping, the inverse mapping can be naturally obtained by transposing them.

3.2. Application to Communication-efficient FL

In the following, we introduce how our mapping, outlined in Sec. 3.1, is utilized for communication-efficient FL.

Mapping construction and broadcasting. We first let the gradient of full model Δw be expressed by the stack of the gradient *w.r.t.* weight matrices for all L layers, i.e., $\Delta w = [dW_1; dW_2; \dots; dW_L]$. Then we define a mapping operation for the full gradient $\varphi(\Delta w)$ as follows:

$$\Delta_\varphi w := \varphi(\Delta w) = [\varphi_1(dW_1); \dots; \varphi_L(dW_L)], \tag{5}$$

where $\varphi_l(\cdot)$ is the mapping function obtained from Eq. (2) in each layer. That is, the mapping $\varphi_l(\cdot)$ is applied in layer-wise manner. Since the structures of the input activation and output gradient change drastically throughout the FL training, the mapping must be reconstructed as the learning progresses. For every specified round of $t \in \mathcal{P}$, where $\mathcal{P}$ is a predefined set of rounds for mapping construction, the server computes the mapping and broadcasts it to all the participating clients. This ensures a synchronization of an up-to-date mapping function being used across all the clients and the server. Note that the server utilizes public data, that is assumed to be publicly available, in constructing mapping. A detailed discussion on public data will follow shortly.

Client-side local updates. For a specific round of t, each n-th client's model is initialized with the t-th global weight as $w_n^{t,0} \leftarrow w^t$. Then the client updates its model $w_n^{t,0}$ with H iterations on the local dataset D_n as follows,

$$w_n^{t,i+1} = w_n^{t,i} - \eta_t \tilde{\nabla} F_n(w_n^{t,i}) \text{ for } i = 0, \dots, H-1 \tag{6}$$

where $\tilde{\nabla} F_n(w_n^t)$ is the stochastic gradient computed via mini-batch SGD, which is an unbiased estimator of the true gradient $\nabla F_n(w_n^t)$. After the local updates as in Eq. (6), the client compute its local gradient as $\Delta w_n^t = w_n^{t,H} - w^t$, i.e., the difference between before and after the local training.

Gradient mapping and compression. After the local gradient is computed, the client transfers the gradient to alternative space through a given mapping function $\varphi(\cdot)$ as $\varphi(\Delta w_n^t)$ which could induce a sparse gradient. Then the client performs the compression in this mapping space. Here, since our chosen compressors, e.g., Top-k and k-Sparse-Binary, are not unbiased estimators, the error accumulated between the compressed and the original gradient is unavoidable even in expectation, which may hinder the reliable convergence behavior. To compensate this, we employ a well-known technique, called error feedback, which has been prevalent in many recent studies (Stich et al., 2018; Wu et al., 2018; Karimireddy et al., 2019; Basu et al., 2019; Li & Li, 2023). To correct this bias, we reflect the error that is accumulated so far before applying compression: $\Delta_\varphi w_n^t = \varphi(\Delta w_n^t) + e_n^t$, and subsequently we accumulate the error again as $e_n^{t+1} = \Delta_\varphi w_n^t - \mathcal{C}(\Delta_\varphi w_n^t)$. After that, we compress $\Delta_\varphi w_n^t$ as $\mathcal{C}(\Delta_\varphi w_n^t)$ and upload it to the server.

Server-side aggregation. Upon receiving the gradients $\mathcal{C}(\Delta_\varphi w_n^t)$ sent from the clients, the server aggregates all these received gradients as follows:

$$\Delta_\varphi w^t = \sum_{n=1}^{N} \kappa_n \mathcal{C}(\Delta_\varphi w_n^t), \tag{7}$$

where κ_n is defined as $|D_n| / \sum_{i=1}^N |D_i|$. The aggregation becomes a simple average when all the clients have the same dataset size, i.e., $\kappa_n = 1/N$. After the aggregated gradient is broadcast, it is inversely mapped back to the original space, i.e., $\Delta w^t = \varphi^{-1}(\Delta_\varphi w^t)$ as $\Delta_\varphi w^t$ is the gradient that is expressed by means of the new mapping space. Then the clients update the t-th round model by the following update rule: $w^{t+1} \leftarrow w^t + \Delta w^t$. This process repeats until the global model converges. The pseudo code of our approach is provided in Algorithm 1.

Algorithm 1 An employment of our mapping in FL

```
1: Input: Initialized weight $w^0$, error $e_n^0 = \mathbf{0}$ for each client $n = 1, 2, \dots N$, compressor $\mathcal{C}(\cdot)$, mapping function $\varphi(\cdot)$ that is constructed and broadcast from the server.
2: for $t = 0, 1, \dots, T-1$ do
3:   for $n \in \{1, 2, \dots, N\}$ in parallel do
4:     $w_n^{t,0} \leftarrow w^t$
5:     for $i = 0, 1, \dots, H-1$ do
6:       $w_n^{t,i+1} = w_n^{t,i} - \eta_t \widetilde{\nabla} F_n(w_n^{t,i})$
7:     end for
8:     $\Delta w_n^t = w_n^{t,H} - w^t$   // Obtaining local gradient
9:     $\Delta_\varphi w_n^t = \varphi(\Delta w_n^t) + e_n^t$   // Mapping & error feedback
10:    Compress $\Delta_\varphi w_n^t$ as $\mathcal{C}(\Delta_\varphi w_n^t)$
11:    Upload $\mathcal{C}(\Delta_\varphi w_n^t)$ to the server
12:    $e_n^{t+1} = \Delta_\varphi w_n^t - \mathcal{C}(\Delta_\varphi w_n^t)$   // Error accumulation
13:  end for
14:  (Server-side) $\Delta_\varphi w^t = \sum_{n=1}^N \kappa_n \mathcal{C}(\Delta_\varphi w_n^t)$
15:  (Server-side) Broadcast $\Delta_\varphi w^t$ to all the clients
16:  for $n \in \{1, 2, \dots, N\}$ and server in parallel do
17:    $\Delta w^t = \varphi^{-1}(\Delta_\varphi w^t)$
18:    $w^{t+1} = w^t + \Delta w^t$
19:  end for
20: end for
```

Remark (Public data). Since the construction of mapping function needs computations of input activation and output gradient as in Eq (2), this process requires the use of data at the server. To this end, we employ the notion of public data, i.e., some open source data publicly available at the server, which has been widely adopted in many recent works (Li & Wang, 2019; Chen & Chao, 2020; Park et al., 2021; Huang et al., 2022; Fang & Ye, 2022; Cho et al., 2022). However, since utilizing public data along with their labels can pose additional difficulties in practice, we adopt a milder assumption where the labels are not available, aiming to relieve this limitation. The server can simply obtain the output gradient from the surrogate cross entropy loss where one-hot vector is replaced by the output probability, i.e., softmax of the output logit for each data. In our simulations, we set only 1% of the entire dataset as the (unlabeled) public data, which is shown to be sufficient for constructing the effective mapping. Finally, we would like to highlight that the requirement of public data at the server is not an issue in traditional distributed learning settings where privacy is less of a concern. Our approach can naturally adapt to this setup by choosing a small portion of training dataset for constructing the mapping before partitioning the data.

4. Convergence Analysis

In this section, we analyze the convergence behavior of our mapping approach for communication-efficient FL.

Assumption 4.1. $F_n(\cdot)$ is an L-smooth function for each n, i.e., $\|\nabla F_n(u) - \nabla F_n(w)\| \leq L\|u - w\|$ for any u and w.

Assumption 4.2. The variance of the stochastic gradient is bounded for each n, i.e., $\mathbb{E}[\|\nabla F_n(w) - \widetilde{\nabla} F_n(w)\|^2] \leq \sigma_n^2$, and the global variance of the local gradient is bounded, i.e, $\frac{1}{N}\sum_{n=1}^N \|\nabla F_n(w) - \nabla F(w)\|^2 \leq \sigma_g^2$ for any w.

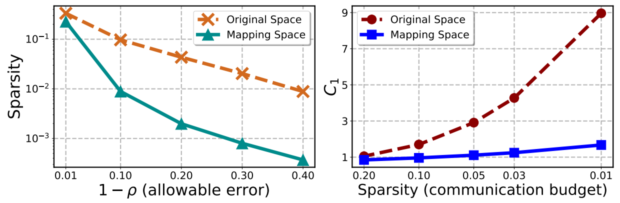


Figure 3: A figure illustrating the trade-off between compression error and sparsity **(left)**, and constant C_1 associated with each sparsity level **(right)** in the original and mapping space.

Assumptions 4.1 and 4.2 are standard in a wide range of FL literature (Li & Li, 2023; Li et al., 2023; Yang et al., 2021b; Reddi et al., 2021; Yang et al., 2021a; Wang et al., 2019). It is worth noting that our analysis does not rely on a bounded gradient assumption, which may not be realistic in practice. Now we are ready to state the following result.

Theorem 4.3. *Let $F_n(w)$ adhere to assumptions 4.1 and 4.2, and suppose compressor $\mathcal{C}(\cdot)$ induces a compression error to the extent of at most $1-\rho$ in terms of l_2 norm, i.e., $\|g - \mathcal{C}(g)\|^2 \leq (1-\rho)\|g\|^2$. Let $\sigma_l^2 = \max_{n\in[N]}\{\sigma_n^2\}$ and $C_1 = \frac{3}{4} + \frac{8(1-\rho)}{\rho^2}$. Then if we choose the learning rate that satisfies $\eta_t \leq \frac{1}{\sqrt{144 C_1} HL}$ in Algorithm 1, it holds that*

$$\frac{1}{\Omega_T}\sum_{t=0}^{T-1} \eta_t \mathbb{E}[\|\nabla F(w^t)\|^2] \lesssim \frac{F(w^0) - F(w^*)}{\Omega_T H} + \frac{L\sigma_l^2}{2\Omega_T N}\sum_{t=0}^{T-1}\eta_t^2 + \frac{3C_1 HL^2(6H\sigma_g^2 + \sigma_l^2)}{\Omega_T}\sum_{t=0}^{T-1}\eta_t^3 \tag{8}$$

where $w^* = \arg\min_{w\in\mathbb{R}^d} F(w)$ and $\Omega_T = \sum_{t=0}^{T-1}\eta_t$.

The proof can be found in Appendix A.1. As can be seen, the convergence bound in Eq. (8) depends on constant C_1 in the third term, which is affected by the compression error $1-\rho$. The constant C_1 takes its minimum value when there is no compression error, i.e, $\rho = 1$, and becomes larger as ρ gets close to 0. Thus, achieving more aggressive compression under a certain allowable error of $1-\rho$ results in greater communication efficiency while maintaining the same convergence bound. As shown in Figure 3, our mapping space is indeed tailored to achieve enhanced compression rate within some specific tolerable error. Conversely, our mapping exhibits lower C_1 within the same communication budget, which leads to a more favorable convergence bound than the original space.

Note that if we choose the learning rate η_t such that $\sum_{t=0}^{T-1}\eta_t \to \infty$ and $\sum_{t=0}^{T-1}\eta_t^2 < \infty, \sum_{t=0}^{T-1}\eta_t^3 < \infty$ as T grows (e.g. $\eta_t = \frac{\eta_0}{c+t}$ for some constant $c > 0$), the right hand side of Eq (8) becomes 0 as $T \to \infty$. This reveals that our result in Theorem 4.3 guarantees the convergence to a stationary point. Next, the following corollary gives an asymptotic rate for a pre-specified number of rounds T.

Corollary 4.4. *Under conditions in Theorem 4.3, for a fixed learning rate $\eta_t = \eta$ chosen to satisfy $\eta = \Theta\left(\sqrt{\frac{N}{TH}}\right)$ for a* *given number of rounds T, the convergence of Algorithm 1 achieves the rate $\mathcal{O}\big(\frac{1}{\sqrt{NTH}}\big)$ as long as $H = \mathcal{O}\big(\frac{T^{1/3}}{N}\big)$.*

As per Corollary 4.4, Algorithm 1 achieves a linear speedup with respect to the number of clients, which matches the existing, best known convergence result of FedAvg. In other words, our mapping does not compromise the convergence speed in spite of its effectiveness. Furthermore, we would like to highlight that our result of linear speedup does not rely on the assumption on the compression of the aggregated gradients (Li & Li, 2023; Haddadpour et al., 2021; Alistarh et al., 2018), which has been made in previous studies (see Assumption 3 in Li & Li, 2023) that utilize the error feedback to guarantee the rate of $\mathcal{O}\big(\frac{1}{\sqrt{NTH}}\big)$. More detailed statement of Corollary 4.4 is provided in Appendix A.1.

5. Related Works

Approaches to reducing communication frequency in FL. There has been a surge in the development of various communication-efficient distributed/federated learning algorithms in response to the demands for minimizing the heavy communication load. One type of technique among these works aims to reduce communication frequency by updating the model with multiple local iterations on each device before uploading the model or gradient to the central server (McMahan et al., 2017; Yu et al., 2019; Wang & Joshi, 2019; Haddadpour et al., 2019). Compared to these works, we take an orthogonal approach by sparsifying the gradient during FL based on the proposed mapping function.

Sparsification & quantization-based methods. A wide range of gradient/model compression methods have been suggested (Basu et al., 2019; Shi et al., 2019; Shlezinger et al., 2020; Qiao et al., 2021; Vargaftik et al., 2022) to improve communication efficiency. Sparsification-based compressors select only a small subset of the elements in the gradient or the model, e.g., based on their absolute magnitudes, to reduce the total amount of communication bits to be transmitted (Wangni et al., 2018; Wang et al., 2018; Shi et al., 2019; Ozfatura et al., 2021; Shah & Lau, 2021; Jiang et al., 2022; Feng et al., 2023; Wang et al., 2023). On the other hand, quantization-based compression methods reduce the bit-width used in representing and transmitting the gradient/model elements, and have been also combined with sparsification-based compressors to pursue heavier compression (Alistarh et al., 2017; Wen et al., 2017; Sattler et al., 2019; Reisizadeh et al., 2020; Jhunjhunwala et al., 2021; Hönig et al., 2022; Qu et al., 2022a; Li & Li, 2023). Although not limited to sparsification or quantization compressor, there have been recent works where unbiased compression is applied for the case of convex and non-convex objectives (Condat et al., 2023; Grudzień et al., 2023; Condat et al., 2024; Gorbunov et al., 2021; Tyurin & Richtárik, 2023; Haddadpour et al., 2021). While all these works conduct compression in the original space, we propose to compress the gradient in the new space that facilitates lossless sparsification. Later in Section 6, we show that the performance of well-known compression methods, specifically Top-k and Sparse-Binary, can be significantly improved when used in conjunction with the proposed mapping function thanks to its lossless sparsifying nature.

Other types of approaches. Beyond sparsification and quantization, different strategies that utilize low-rank compression, countsketch, and other techniques (Li et al., 2021a; Sery et al., 2021) have been also proposed. Low-rank based communication-efficient methods decompose the weight of each layer into smaller matrices, or starts training from decomposed (smaller) ones, which downsizes the total size of the gradient to be communicated (Hyeon-Woo et al., 2022; Konečnỳ et al., 2016; Qiao et al., 2021). We would like to highlight that instead of leveraging the structure of the weight or decomposing it as done in these works, we make use of the structures of the input activation and the output gradient, both of which are explicitly related to the gradient itself. Additionally, countsketch-based compressors reduce or project the original dimension of computed gradient into the smaller dimension, from which the elements with large magnitudes can later be recovered (Rothchild et al., 2020; Ivkin et al., 2019). In these approaches, however, either an additional assumption or a second round of communication is needed to guarantee the convergence of FL. On the contrary, our mapping approach maintains the existing convergence rate without requiring any further conditions on the gradient or the model other than the standard assumptions.

In addition to these works, there have been recent papers that share a conceptual similarity with our work. Specifically, (Qian et al., 2022) aimed to identify a new set of vectors that allows the Hessian to be expressed using a smaller number of basis vectors, thereby enhancing its compressibility, which is crucial for the Newton's method in the problem of Generalized Linear Model (GLM). Similarly, in (Safaryan et al., 2021; Wang et al., 2022), they utilize what is known as a smoothness matrix, multiplying it with the computed gradient to make it a more compressible vector. Compared to these works, our work considers a new set of vectors to represent the element of the mapping space for *each layer* by utilizing the input vectors. This makes our method applicable to a broader range of modern architectures like complex DNNs with arbitrary non-convex objectives. Our approach can be adopted to any structure that turns out to be linear operation, which allows us to implement our method to various types of modules, e.g., attention layer in the transformer architectures. Moreover, based on the gradient derivation, we are able to find a more effective mapping by considering not only the input activation but also the output gradient, which further maximizes the compressibility.

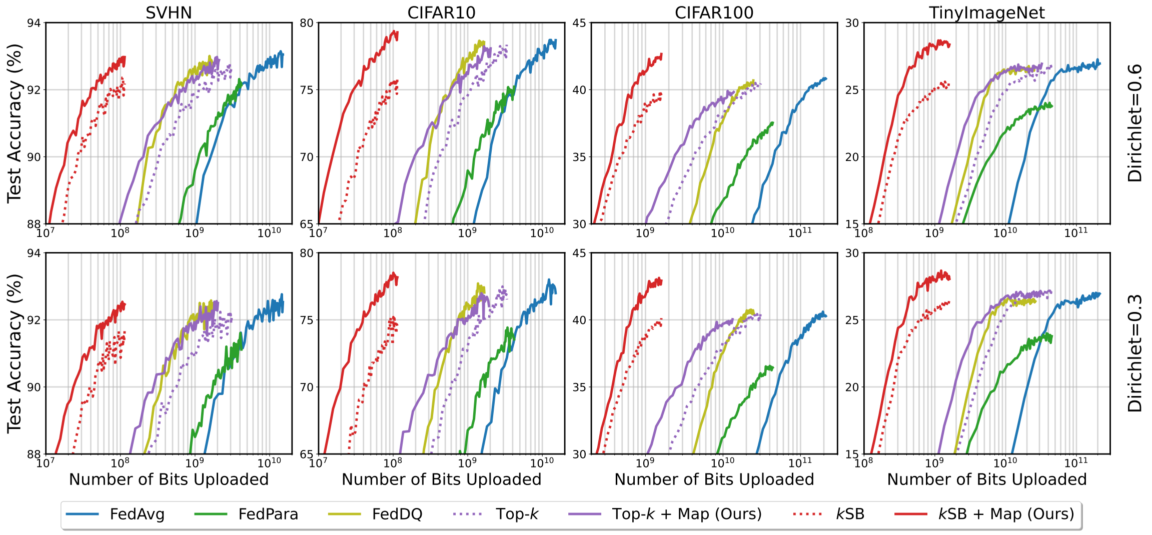


Figure 4: Comparative analysis of accuracy versus number of communication bits across various benchmarks, including SVHN, CIFAR10, CIFAR100 and TinyImageNet under Dirichlet distribution with $\alpha = 0.6$ **(top row)** and $\alpha = 0.3$ **(bottom row)** among clients. Note that both Top-k and kSB improve significantly when combined with our mapping scheme (+Map).

6. Experiments

In this section, we evaluate our proposed mapping approach in terms of communication efficiency in FL setup.

6.1. Experimental Details

We simulate FL training where the number of clients is set to be $N = 100$, with the contact ratio being 0.1, that is, 10 clients are randomly sampled for communication at each FL round. We evaluate our method in comparison with other baselines on four benchmark datasets: SVHN, CIFAR10, CIFAR100, and TinyImageNet. We essentially follow the settings outlined in previous FL works (Gao et al., 2022; Acar et al., 2021). We investigate a non-IID data setup, and we configure the data split to follow the Dirichlet distribution with parameter values $\alpha = 0.6$ and $\alpha = 0.3$. Data heterogeneity gets more pronounced as α decreases. The dataset size is balanced across all clients. For the SVHN and CIFAR10 datasets, we use a CNN model, which consists of two 5×5 convolution layers followed by three fully-connected layers, and as for the CIFAR100 and TinyImageNet datasets, we use the ResNet18 model. In the construction of mapping, we randomly sample and designate 1% of the entire dataset as the unlabeled public data. We then split the remaining data to be assigned to the clients. To validate the effectiveness of our approach, we apply our mapping to two types of sparsification-based compressors, namely, Top-k and k-Sparse-Binary to fully reap the benefit of the lossless sparsifiying nature of our mapping. When applying Top-k and k-Sparse-Binary in our mapping space, the accumulated error for the error feedback is initialized to zero whenever the mapping is newly constructed to account for the staleness of near-zero elements in the mapping space. More detailed design parameters including periods for mapping construction are provided in Appendix A.2.

Baselines. We compare our approach with various communication efficient FL baselines: Top-k, k-Sparse-Binary (kSB), FedDQ (Qu et al., 2022a) and FedPara (Hyeon-Woo et al., 2022). To make a fair comparison, we design two additional baselines that utilize the public data. We relieve the assumption in that the public data can also be accessed by the clients, so that all the participating clients can make prediction on a given shared public data. Each client then uploads its predicted output probability along with the local update at each round. There are two possible approaches here: one involves incorporating the gradient computed from distilled loss to the aggregated gradient in Eq. (7), and the other is to train the model on public data in the server right after the model update with the aggregated gradient, which we denote PD1 and PD2, respectively. We apply PD1 and PD2 to both Top-k and Sparse-Binary as baselines. See Appendix A.5 for more details.

As for Top-k and its counterpart applied with mapping (denoted by 'Top-k + Map'), we evaluate these methods across four different levels of compression error, $1 - \rho$, i.e., $\rho = \{0.9, 0.8, 0.7, 0.6\}$. This can be done by sparsifying the smallest elements until the desired value of ρ is reached. For kSB and kSB + Map, as it is difficult to directly control the compression error due to subsequent quantization process, we control the sparsity s with four different levels, i.e., $s = \{0.1, 0.05, 0.03, 0.01\}$. We present the case of $\rho = 0.7$ for Top-k and $s = 0.03$ for kSB in the main results. Results for other values of ρ and s are provided in Appendix A.12.

6.2. Main Results

Mapping improves communication efficiency. We first provide comparative analysis of accuracy versus number of bits per client during uplink transmission. Following

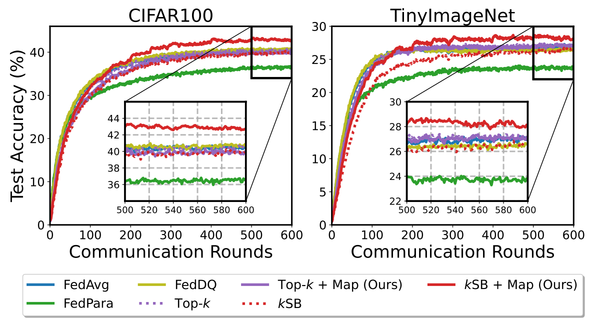


Figure 5: Comparative figures of accuracy versus communication rounds on CIFAR100 and TinyImageNet in the case of $\alpha = 0.3$.

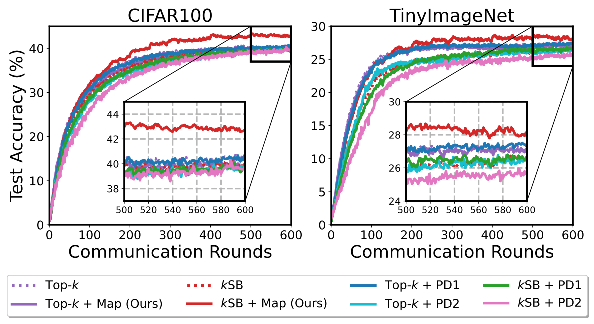


Figure 6: Comparison results of accuracy versus communication rounds with PD baselines in the case of $\alpha = 0.3$.

the prior works (Isik et al., 2023; Li & Li, 2023; Vargaftik et al., 2022; Qu et al., 2022a; Haddadpour et al., 2021; Bernstein et al., 2018; Wen et al., 2017), we focus on the uplink communication load, which is the main bottleneck in FL (Kairouz et al., 2021). As can be seen in Figure 4, our mapping approach (solid lines) improves the existing sparsification based methods, Top-k and kSB (dotted lines) in terms of communication efficiency across all the datasets. Although FedDQ slightly outperforms Top-k + Map on the CIFAR10 and CIFAR100 datasets, our mapping makes the existing Top-k scheme either comparable to or outperform the baselines on SVHN and TinyImageNet. Moreover, kSB + Map surpasses other baselines with large gaps across all datasets, which is not the case for the naive kSB.

SB reaches higher accuracy than vanilla FedAvg. We provide comparison results across communication rounds, which shows selective results on CIFAR100 and TinyImageNet due to space limitation. As shown in Figure 5, kSB + Map achieves higher accuracy, even than the conventional FedAvg where the uncompressed gradient is uploaded. This additional benefit may partly be attributed to the ability of the quantization-based compressors to reduce the deviation among local updates induced by data heterogeneity, as can seen in (Oh et al., 2023). One notable aspect of our mapping space is the concentration of most of the information into a smaller number of elements in a lossless manner before compression. As such, the mapping space can preserve a greater amount of information than the original space while

Table 1: The amount of bits (GiB) required to reach the desired target accuracy of 78% for CIFAR10 and 41% for CIFAR100.

Dataset	**FedAvg**	**LMap**	**HMap**
CIFAR10	1.014	0.0068 ($\times$**149.1**)	0.0075 ($\times$**135.2**)
CIFAR100	23.09	0.0940 ($\times$**245.6**)	0.0899 ($\times$**256.8**)

Table 2: The amount of bits required to reach the target accuracy of 41% for CIFAR100 across various ratios of public data.

Ratio of public data	0.1%	0.5%	1%	5%
Bits (GiB)	0.0956	0.0903	0.0940	0.0944

sparsifying it. This property of our mapping space, coupled with the quantizer's ability to lessen the effects of data heterogeneity, leads to better accuracy than vanilla FedAvg.

PD baselines fail to enhance Top-k and kSB. Lastly, we compare our approach with more fair baselines, Top-k and kSB that utilize the public data (we denote these baselines as '+ PD1' and '+ PD2'), as mentioned above. As can be seen in Figure 6, incorporating public data fails to enhance the performance of the existing Top-k and kSB schemes due to the limited availability of the public data. Given this, despite utilizing the same quantity of public data, our mapping approach is the only strategy that uniquely succeeds in improving the existing method, especially the kSB compressor. Additional results on varying values of s, ρ, α and other datasets are supplemented in Appendix A.13.

6.3. Further Studies

Robustness to heterogeneous public data. In this study, we investigate the robustness of our mapping to the degree of heterogeneity in public data. We evaluate on the CIFAR10 and CIFAR100 datasets under $\alpha = 0.6$ using the kSB compressor. To simulate scenarios with heterogeneous public data, we sample from CIFAR10 dataset and use the samples as public data when evaluating on CIFAR100, and vice versa. Table 1 shows the number of bits required to achieve desired target accuracy for both **L**ow heterogeneous **Map** (denoted by 'LMap', which is the map used so far) and **H**igh heterogeneous **Map** (denoted by 'HMap', which represents the map created from a different dataset). As can be seen, HMap shows enhanced communication efficiency just as good as LMap. It achieves an efficiency of $\times 256.8$ to reach target accuracy on CIFAR100 compared to naive FedAvg. These results indicate that our mapping can provide a sufficiently effective new space, even when constructed with public data from different classes, as long as the domains are similar.

Ablation on the amount of public data. Here we conduct an ablation study on varying amounts of public data used in constructing mapping. We evaluate our approach in the ablation on CIFAR100 under $\alpha = 0.6$ using the kSB compressor across different ratios of public data from 0.1% to 5% of the entire dataset. For the case of 5%, we additionally sample the public data from the remaining data after distributing them, to ensure consistency in data configurations among clients cross ratios. As can be seen in Table 2, our mapping still yields reasonable results, even when constructed with a very small amount of public data, specifically 0.1%. This indicates that a very limited quantity of data suffices to establish an effective mapping space.

Extra costs for mapping function. Although our focus is on the uplink transmission as done in many previous works, there are extra costs incurred in both (downlink) communication and (server-side) computation when employing the mapping function. Our mapping allows a trade-off by offloading the heavy communication burden on the client-side to the server. We wish to highlight that this is a beneficial trade-off in practice, since the server typically has a large computing power and downlink tends to have a higher communication rate compared to the uplink transmission. We have further elaborated on this issue in Appendix A.6.

Further considerations. There can be other considerations including time duration for mapping, the effects of the periods for mapping reconstruction on the performance, and sparse encoding strategies. We have further discussed these aspects in Appendix A.7, A.8 and A.9.

7. Conclusion

In this paper, we presented a novel approach to mitigating heavy communication load in federated learning scenarios. We move beyond the original space and propose to construct a mapping function that can provide more sparsified gradients in a lossless manner. In light of this, applying our mapping to sparsification-based compressors can achieve more aggressive compression with minimal information loss, leading to enhanced communication efficiency during FL, which is also verified by our theoretical analysis. Unlike other baselines in the original space, our method often reaches higher accuracy than conventional FedAvg, which is an additional benefit unique to our mapping space.

Acknowledgements

This work was supported by Samsung Electronics Co., Ltd., by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. NRF-2019R1I1A2A02061135), and by IITP funds from MSIT of Korea (No. 2020-0-00626).

Impact Statement

This work presents a significant advancement in the field of communication-efficient FL, particularly in reducing uplink transmission, through a novel mapping approach towards aggressive gradient sparsification.

Although the local data is not exchanged directly, there still remains a risk of privacy leakage to a certain extent due to the uploading of weight update. This risk can be mitigated by employing existing differential privacy techniques, which adds some noise to data or weight update. Moreover, ensuring that the proposed method does not unintentionally bias the model or contribute to unfair outcomes is crucial. This concern might also be addressed by existing approaches suggested in the field of machine learning fairness.

References

Acar, D. A. E. and Saligrama, V. Fedhen: Federated learning in heterogeneous networks. *arXiv preprint arXiv:2207.03031*, 2022.

Acar, D. A. E., Zhao, Y., Matas, R., Mattina, M., Whatmough, P., and Saligrama, V. Federated learning based on dynamic regularization. In *International Conference on Learning Representations*, 2021.

Alistarh, D., Grubic, D., Li, J., Tomioka, R., and Vojnovic, M. Qsgd: Communication-efficient sgd via gradient quantization and encoding. *Advances in neural information processing systems*, 30, 2017.

Alistarh, D., Hoefler, T., Johansson, M., Konstantinov, N., Khirirat, S., and Renggli, C. The convergence of sparsified gradient methods. *Advances in Neural Information Processing Systems*, 31, 2018.

Antunes, R. S., André da Costa, C., Küderle, A., Yari, I. A., and Eskofier, B. Federated learning for healthcare: Systematic review and architecture proposal. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 13 (4):1–23, 2022.

Aurna, N. F., Hossain, M. D., Taenaka, Y., and Kadobayashi, Y. Federated learning-based credit card fraud detection: Performance analysis with sampling methods and deep learning algorithms. In *2023 IEEE International Conference on Cyber Security and Resilience (CSR)*, pp. 180–186. IEEE, 2023.

Basu, D., Data, D., Karakus, C., and Diggavi, S. Qsparse-local-sgd: Distributed sgd with quantization, sparsification and local computations. *Advances in Neural Information Processing Systems*, 32, 2019.

Bernstein, J., Wang, Y.-X., Azizzadenesheli, K., and Anandkumar, A. signsgd: Compressed optimisation for non-convex problems. In *International Conference on Machine Learning*, pp. 560–569. PMLR, 2018.

Bonawitz, K., Eichner, H., Grieskamp, W., Huba, D., Ingerman, A., Ivanov, V., Kiddon, C., Konečnỳ, J., Mazzocchi, S., McMahan, B., et al. Towards federated learning at scale: System design. *Proceedings of machine learning and systems*, 1:374–388, 2019.

Chen, H.-Y. and Chao, W.-L. Fedbe: Making bayesian model ensemble applicable to federated learning. *arXiv preprint arXiv:2009.01974*, 2020.

Cho, Y. J., Manoel, A., Joshi, G., Sim, R., and Dimitriadis, D. Heterogeneous ensemble knowledge transfer for training large models in federated learning. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2022.

Condat, L., Agarskỳ, I., Malinovsky, G., and Richtárik, P. Tamuna: Doubly accelerated federated learning with local training, compression, and partial participation. *arXiv preprint arXiv:2302.09832*, 2023.

Condat, L., Maranjyan, A., and Richtarik, P. Locodl: Communication-efficient distributed learning with local training and compression. *arXiv preprint arXiv:2403.04348*, 2024.

Fang, X. and Ye, M. Robust federated learning with noisy and heterogeneous clients. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10072–10081, 2022.

Feng, Z., Chen, X., Wu, Q., Wu, W., Zhang, X., and Huang, Q. Feddd: Toward communication-efficient federated learning with differential parameter dropout. *IEEE Transactions on Mobile Computing*, 2023.

Gao, L., Fu, H., Li, L., Chen, Y., Xu, M., and Xu, C.-Z. Feddc: Federated learning with non-iid data via local drift decoupling and correction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10112–10121, 2022.

Gorbunov, E., Burlachenko, K. P., Li, Z., and Richtárik, P. Marina: Faster non-convex distributed learning with compression. In *International Conference on Machine Learning*, pp. 3788–3798. PMLR, 2021.

Grudzień, M., Malinovsky, G., and Richtárik, P. Improving accelerated federated learning with compression and importance sampling. *arXiv preprint arXiv:2306.03240*, 2023.

Haddadpour, F., Kamani, M. M., Mahdavi, M., and Cadambe, V. Local sgd with periodic averaging: Tighter analysis and adaptive synchronization. *Advances in Neural Information Processing Systems*, 32, 2019.

Haddadpour, F., Kamani, M. M., Mokhtari, A., and Mahdavi, M. Federated learning with compression: Unified analysis and sharp guarantees. In *International Conference on Artificial Intelligence and Statistics*, pp. 2350–2358. PMLR, 2021.

Hönig, R., Zhao, Y., and Mullins, R. Dadaquant: Doubly-adaptive quantization for communication-efficient federated learning. In *International Conference on Machine Learning*, pp. 8852–8866. PMLR, 2022.

Horváth, S. and Richtarik, P. A better alternative to error feedback for communication-efficient distributed learning. In *International Conference on Learning Representations*, 2021.

Huang, W., Ye, M., and Du, B. Learn from others and be yourself in heterogeneous federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10143–10153, 2022.

Hyeon-Woo, N., Ye-Bin, M., and Oh, T.-H. Fedpara: Low-rank hadamard product for communication-efficient federated learning. In *International Conference on Learning Representations*, 2022.

Isik, B., Pase, F., Gunduz, D., Weissman, T., and Michele, Z. Sparse random networks for communication-efficient federated learning. In *The Eleventh International Conference on Learning Representations*, 2023.

Ivkin, N., Rothchild, D., Ullah, E., Stoica, I., Arora, R., et al. Communication-efficient distributed sgd with sketching. *Advances in Neural Information Processing Systems*, 32, 2019.

Jhunjhunwala, D., Gadhikar, A., Joshi, G., and Eldar, Y. C. Adaptive quantization of model updates for communication-efficient federated learning. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3110–3114. IEEE, 2021.

Jiang, Y., Wang, S., Valls, V., Ko, B. J., Lee, W.-H., Leung, K. K., and Tassiulas, L. Model pruning enables efficient federated learning on edge devices. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021.

Karimireddy, S. P., Rebjock, Q., Stich, S., and Jaggi, M. Error feedback fixes signsgd and other gradient compression schemes. In *International Conference on Machine Learning*, pp. 3252–3261. PMLR, 2019.

Kim, D.-Y., Han, D.-J., Seo, J., and Moon, J. Warping the space: Weight space rotation for class-incremental few-shot learning. In *The Eleventh International Conference on Learning Representations*, 2023.

Konečný, J., McMahan, H. B., Yu, F. X., Richtárik, P., Suresh, A. T., and Bacon, D. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

Li, C., Li, G., and Varshney, P. K. Communication-efficient federated learning based on compressed sensing. *IEEE Internet of Things Journal*, 8(20):15531–15541, 2021a.

Li, D. and Wang, J. Fedmd: Heterogenous federated learning via model distillation. *arXiv preprint arXiv:1910.03581*, 2019.

Li, J., Meng, Y., Ma, L., Du, S., Zhu, H., Pei, Q., and Shen, X. A federated learning based privacy-preserving smart healthcare system. *IEEE Transactions on Industrial Informatics*, 18(3), 2021b.

Li, Q., Zhang, M., Yin, N., Yin, Q., and Shen, L. Asymmetrically decentralized federated learning. *arXiv preprint arXiv:2310.05093*, 2023.

Li, T., Sahu, A. K., Talwalkar, A., and Smith, V. Federated learning: Challenges, methods, and future directions. *IEEE signal processing magazine*, 37(3):50–60, 2020a.

Li, X. and Li, P. Analysis of error feedback in federated non-convex optimization with biased compression: Fast convergence and partial participation. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 19638–19688. PMLR, 23–29 Jul 2023.

Li, X., Huang, K., Yang, W., Wang, S., and Zhang, Z. On the convergence of fedavg on non-iid data. In *International Conference on Learning Representations*, 2020b.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pp. 1273–1282. PMLR, 2017.

Nguyen, A., Do, T., Tran, M., Nguyen, B. X., Duong, C., Phan, T., Tjiputra, E., and Tran, Q. D. Deep federated learning for autonomous driving. In *2022 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1824–1830. IEEE, 2022.

Oh, Y., Jeon, Y.-S., Chen, M., and Saad, W. Fedvqcs: Federated learning via vector quantized compressed sensing. *IEEE Transactions on Wireless Communications*, 2023.

Ozfatura, E., Ozfatura, K., and Gündüz, D. Time-correlated sparsification for communication-efficient federated learning. In *2021 IEEE International Symposium on Information Theory (ISIT)*, pp. 461–466. IEEE, 2021.

Park, J., Han, D.-J., Choi, M., and Moon, J. Sageflow: Robust federated learning against both stragglers and adversaries. *Advances in neural information processing systems*, 34:840–851, 2021.

Qian, X., Islamov, R., Safaryan, M., and Richtarik, P. Basis matters: Better communication-efficient second order methods for federated learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 680–720. PMLR, 2022.

Qiao, Z., Yu, X., Zhang, J., and Letaief, K. B. Communication-efficient federated learning with dual-side low-rank compression. *arXiv preprint arXiv:2104.12416*, 2021.

Qu, L., Song, S., and Tsui, C.-Y. Feddq: Communication-efficient federated learning with descending quantization. In *GLOBECOM 2022-2022 IEEE Global Communications Conference*, pp. 281–286. IEEE, 2022a.

Qu, L., Zhou, Y., Liang, P. P., Xia, Y., Wang, F., Adeli, E., Fei-Fei, L., and Rubin, D. Rethinking architecture design for tackling data heterogeneity in federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10061–10071, 2022b.

Reddi, S. J., Charles, Z., Zaheer, M., Garrett, Z., Rush, K., Konečný, J., Kumar, S., and McMahan, H. B. Adaptive federated optimization. In *International Conference on Learning Representations*, 2021.

Reisizadeh, A., Mokhtari, A., Hassani, H., Jadbabaie, A., and Pedarsani, R. Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization. In *International Conference on Artificial Intelligence and Statistics*, pp. 2021–2031. PMLR, 2020.

Rothchild, D., Panda, A., Ullah, E., Ivkin, N., Stoica, I., Braverman, V., Gonzalez, J., and Arora, R. Fetchsgd: Communication-efficient federated learning with sketching. In *International Conference on Machine Learning*, pp. 8253–8265. PMLR, 2020.

Safaryan, M., Hanzely, F., and Richtárik, P. Smoothness matrices beat smoothness constants: Better communication compression techniques for distributed optimization. *Advances in Neural Information Processing Systems*, 34: 25688–25702, 2021.

Sattler, F., Wiedemann, S., Müller, K.-R., and Samek, W. Sparse binary compression: Towards distributed deep learning with minimal communication. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2019.

Sery, T., Shlezinger, N., Cohen, K., and Eldar, Y. C. Over-the-air federated learning from heterogeneous data. *IEEE Transactions on Signal Processing*, 69:3796–3811, 2021.

Shah, S. M. and Lau, V. K. Model compression for communication efficient federated learning. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

Shi, S., Wang, Q., Zhao, K., Tang, Z., Wang, Y., Huang, X., and Chu, X. A distributed synchronous sgd algorithm with global top-k sparsification for low bandwidth networks. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pp. 2238–2247. IEEE, 2019.

Shlezinger, N., Chen, M., Eldar, Y. C., Poor, H. V., and Cui, S. Uveqfed: Universal vector quantization for federated learning. *IEEE Transactions on Signal Processing*, 69: 500–514, 2020.

Stich, S. U., Cordonnier, J.-B., and Jaggi, M. Sparsified sgd with memory. *Advances in Neural Information Processing Systems*, 31, 2018.

Tyurin, A. and Richtárik, P. DASHA: Distributed nonconvex optimization with communication compression and optimal oracle complexity. In *The Eleventh International Conference on Learning Representations*, 2023.

Vargaftik, S., Basat, R. B., Portnoy, A., Mendelson, G., Itzhak, Y. B., and Mitzenmacher, M. Eden: Communication-efficient and robust distributed mean estimation for federated learning. In *International Conference on Machine Learning*, pp. 21984–22014. PMLR, 2022.

Vogels, T., Karimireddy, S. P., and Jaggi, M. Powersgd: Practical low-rank gradient compression for distributed optimization. *Advances in Neural Information Processing Systems*, 32, 2019.

Wang, B., Safaryan, M., and Richtárik, P. Theoretically better and numerically faster distributed optimization with smoothness-aware quantization techniques. *Advances in Neural Information Processing Systems*, 35:9841–9852, 2022.

Wang, B., Fang, J., Li, H., and Zeng, B. Communication-efficient federated learning: A variance-reduced stochastic approach with adaptive sparsification. *IEEE Transactions on Signal Processing*, 2023.

Wang, H., Sievert, S., Liu, S., Charles, Z., Papailiopoulos, D., and Wright, S. Atomo: Communication-efficient learning via atomic sparsification. *Advances in neural information processing systems*, 31, 2018.

Wang, J. and Joshi, G. Adaptive communication strategies to achieve the best error-runtime trade-off in local-update sgd. *Proceedings of Machine Learning and Systems*, 1: 212–229, 2019.

Wang, S., Tuor, T., Salonidis, T., Leung, K. K., Makaya, C., He, T., and Chan, K. Adaptive federated learning in resource constrained edge computing systems. *IEEE journal on selected areas in communications*, 37(6):1205–1221, 2019.

Wangni, J., Wang, J., Liu, J., and Zhang, T. Gradient sparsification for communication-efficient distributed optimization. *Advances in Neural Information Processing Systems*, 31, 2018.

Wen, W., Xu, C., Yan, F., Wu, C., Wang, Y., Chen, Y., and Li, H. Terngrad: Ternary gradients to reduce communication in distributed deep learning. *Advances in neural information processing systems*, 30, 2017.

Wu, J., Huang, W., Huang, J., and Zhang, T. Error compensated quantized sgd and its applications to large-scale distributed optimization. In *International Conference on Machine Learning*, pp. 5325–5333. PMLR, 2018.

Yang, H., Fang, M., and Liu, J. Achieving linear speedup with partial worker participation in non-iid federated learning. In *International Conference on Learning Representations*, 2021a.

Yang, H., Liu, J., and Bentley, E. S. Cfedavg: achieving efficient communication and fast convergence in non-iid federated learning. In *2021 19th International Symposium on Modeling and Optimization in Mobile, Ad hoc, and Wireless Networks (WiOpt)*, pp. 1–8. IEEE, 2021b.

Yu, H., Yang, S., and Zhu, S. Parallel restarted sgd with faster convergence and less communication: Demystifying why model averaging works for deep learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 5693–5700, 2019.

Zhang, H., Bosch, J., and Olsson, H. H. End-to-end federated learning for autonomous driving vehicles. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2021.

Zheng, S., Huang, Z., and Kwok, J. Communication-efficient distributed blockwise momentum sgd with error-feedback. *Advances in Neural Information Processing Systems*, 32, 2019.

Zhu, H., Xu, J., Liu, S., and Jin, Y. Federated learning on non-iid data: A survey. *Neurocomputing*, 465:371–390, 2021.

A. Appendix

A.1. Convergence Analysis

In this section, we analyze the convergence behavior of our approach for communication-efficient federated learning. To this end, we consider the following assumptions.

Assumption A.1. $F_n(\cdot)$ is L-smooth function for each n, i.e., $\|\nabla F_n(u) - \nabla F_n(w)\| \le L\|u - w\|$ for any u and w.

Assumption A.2. The variance of the stochastic gradient is bounded for each n, i.e., $\mathbb{E}[\|\nabla F_n(w) - \widetilde{\nabla} F_n(w)\|^2] \le \sigma_n^2$, and the global variance of the local gradient is bounded, i.e, $\frac{1}{N}\sum_{n=1}^{N}\|\nabla F_n(w) - \nabla F(w)\|^2 \le \sigma_g^2$ for any w.

Moreover, since our mapping $\varphi(\cdot)$ is constructed and defined based on Eq. (2) - (5), we have the following useful properties.

Property A.3. Mapping $\varphi(\cdot) : \mathbb{R}^d \to \mathbb{R}^d$ is a linear transformation, i.e., $\varphi(\alpha u + \beta w) = \alpha\varphi(u) + \beta\varphi(w)$ for any $\alpha, \beta \in \mathbb{R}$ and $v, w \in \mathbb{R}^d$. Its inverse mapping $\varphi^{-1}(\cdot) : \mathbb{R}^d \to \mathbb{R}^d$ follows the same property.

Property A.4. Mapping $\varphi(\cdot) : \mathbb{R}^d \to \mathbb{R}^d$ preserves the l_2 norm, i.e., $\|\varphi(u)\| = \|u\|$ for any $u \in \mathbb{R}^d$. Its inverse mapping $\varphi^{-1}(\cdot) : \mathbb{R}^d \to \mathbb{R}^d$ follows the same property.

Now we start with the following intermediate lemmas, drawing on (Yang et al., 2021a; Reddi et al., 2021; Li & Li, 2023).

Lemma A.5. *Under Assumption A.1 and A.2, for any $i \in [H]$ and $\eta_t \le \frac{1}{4HL}$, it is observed that,*

$$\frac{1}{N}\sum_{n=1}^{N}\mathbb{E}\big[\|w_n^{t,i} - w^t\|^2\big] \le 3H\eta_t^2\left(6H\sigma_g^2 + \frac{1}{N}\sum_{n=1}^{N}\sigma_n^2\right) + 18H^2\eta_t^2\mathbb{E}\big[\|\nabla F(w^t)\|^2\big].$$

Proof. For any $i = 1, \ldots, H$, we have

$$\frac{1}{N}\sum_{n=1}^{N}\mathbb{E}\big[\|w_n^{t,i} - w^t\|^2\big] = \frac{1}{N}\sum_{n=1}^{N}\mathbb{E}\big[\|w_n^{t,i-1} - w^t - \eta_t\widetilde{\nabla} F_n(w_n^{t,i-1})\|^2\big]$$

$$= \frac{1}{N}\sum_{n=1}^{N}\mathbb{E}\Big[\mathbb{E}\big[\|w_n^{t,i-1} - w^t - \eta_t\nabla F_n(w_n^{t,i-1}) + \big(\eta_t\nabla F_n(w_n^{t,i-1}) - \eta_t\widetilde{\nabla} F_n(w_n^{t,i-1})\big)\|^2 | w_n^{t,i-1}\big]\Big]$$

$$= \frac{1}{N}\sum_{n=1}^{N}\mathbb{E}\big[\|w_n^{t,i-1} - w^t - \eta_t\nabla F_n(w_n^{t,i-1})\|^2\big] + \frac{1}{N}\sum_{n=1}^{N}\mathbb{E}\big[\|\eta_t\nabla F_n(w_n^{t,i-1}) - \eta_t\widetilde{\nabla} F_n(w_n^{t,i-1})\|^2\big]$$

$$= \frac{1}{N}\sum_{n=1}^{N}\mathbb{E}\big[\|w_n^{t,i-1} - w^t - \eta_t\nabla F_n(w_n^{t,i-1}) + \eta_t\nabla F_n(w^t) - \eta_t\nabla F_n(w^t) + \eta_t\nabla F(w^t) - \eta_t\nabla F(w^t)\|^2\big]$$

$$+ \underbrace{\frac{1}{N}\sum_{n=1}^{N}\mathbb{E}\big[\|\eta_t\nabla F_n(w_n^{t,i-1}) - \eta_t\widetilde{\nabla} F_n(w_n^{t,i-1})\|^2\big]}_{\le \frac{\eta_t^2}{N}\sum_{n=1}^{N}\sigma_n^2 \text{ via Assumption A.2}}$$

$$\overset{(a)}{\le} \frac{2H}{2H-1}\cdot\frac{1}{N}\sum_{n=1}^{N}\mathbb{E}\big[\|w_n^{t,i-1} - w^t\|^2\big] + \frac{6H}{N}\sum_{n=1}^{N}\mathbb{E}\big[\|\eta_t\nabla F_n(w_n^{t,i-1}) - \eta_t\nabla F_n(w^t)\|^2\big]$$

$$+ \underbrace{\frac{6H}{N}\sum_{n=1}^{N}\mathbb{E}\big[\|\eta_t\nabla F_n(w^t) - \eta_t\nabla F(w^t)\|^2\big]}_{\le 6H\eta_t^2\sigma_g^2 \text{ via Assumption A.2}} + \frac{6H}{N}\sum_{n=1}^{N}\mathbb{E}\big[\|\eta_t\nabla F(w^t)\|^2\big] + \frac{\eta_t^2}{N}\sum_{n=1}^{N}\sigma_n^2$$

$$\le \frac{2H}{2H-1}\cdot\frac{1}{N}\sum_{n=1}^{N}\mathbb{E}\big[\|w_n^{t,i-1} - w^t\|^2\big] + \underbrace{\frac{6H\eta_t^2L^2}{N}\sum_{n=1}^{N}\mathbb{E}\big[\|w_n^{t,i-1} - w^t\|^2\big]}_{\text{Due to Assumption A.1}}$$

$$+ \eta_t^2\left(6H\sigma_g^2 + \frac{1}{N}\sum_{n=1}^{N}\sigma_n^2\right) + 6H\eta_t^2\mathbb{E}\big[\|\nabla F(w^t)\|^2\big]$$

$$= \left(\frac{2H}{2H-1} + 6H\eta_t^2L^2\right)\frac{1}{N}\sum_{n=1}^{N}\mathbb{E}\big[\|w_n^{t,i-1} - w^t\|^2\big] + \eta_t^2\left(6H\sigma_g^2 + \frac{1}{N}\sum_{n=1}^{N}\sigma_n^2\right) + 6H\eta_t^2\mathbb{E}\big[\|\nabla F(w^t)\|^2\big]$$

$$\overset{(b)}{\leq} \left(\frac{2H+1}{2H-1}\right)\frac{1}{N}\sum_{n=1}^{N}\mathbb{E}\big[\|w_n^{t,i-1} - w^t\|^2\big] + \eta_t^2\left(6H\sigma_g^2 + \frac{1}{N}\sum_{n=1}^{N}\sigma_n^2\right) + 6H\eta_t^2\mathbb{E}\big[\|\nabla F(w^t)\|^2\big]$$

where (a) comes from Jensen's inequality and (b) comes from the fact that $6H\eta_t^2L^2 \leq \frac{6}{16H} \leq \frac{1}{2H-1}$ when $\eta_t \leq \frac{1}{4HL}$. By repeating the above derivation while keeping in mind that $w_n^{t,0} = w^t$, we attain

$$\begin{aligned}\frac{1}{N}\sum_{n=1}^{N}\mathbb{E}\big[\|w_n^{t,i} - w^t\|^2\big] &\leq \sum_{j=0}^{i-1}\left(\frac{2H+1}{2H-1}\right)^j\left[\eta_t^2\left(6H\sigma_g^2 + \frac{1}{N}\sum_{n=1}^{N}\sigma_n^2\right) + 6H\eta_t^2\mathbb{E}\big[\|\nabla F(w^t)\|^2\big]\right]\\ &\leq \sum_{j=0}^{H-1}\left(\frac{2H+1}{2H-1}\right)^H\left[\eta_t^2\left(6H\sigma_g^2 + \frac{1}{N}\sum_{n=1}^{N}\sigma_n^2\right) + 6H\eta_t^2\mathbb{E}\big[\|\nabla F(w^t)\|^2\big]\right]\\ &\leq 3H\left[\eta_t^2\left(6H\sigma_g^2 + \frac{1}{N}\sum_{n=1}^{N}\sigma_n^2\right) + 6H\eta_t^2\mathbb{E}\big[\|\nabla F(w^t)\|^2\big]\right]\end{aligned}$$

where the last inequality is due to $\left(\frac{2H+1}{2H-1}\right)^H \leq 3$ whenever $H \geq 1$, which completes the proof. ☐

Lemma A.6. *For a given compressor $\mathcal{C}(\cdot)$ that satisfies $\|g - \mathcal{C}(g)\|^2 \leq (1-\rho)\|g\|^2$ for all $g \in \mathbb{R}^d$, and for any $t \geq 1$, one can verify that,*

$$\frac{1}{N}\sum_{n=1}^{N}\mathbb{E}\big[\|e_n^t\|^2\big] \leq \frac{2(1-\rho)}{\rho}\frac{1}{N}\sum_{n=1}^{N}\sum_{j=0}^{t-1}\left(\frac{2-\rho}{2}\right)^j\mathbb{E}\big[\|\Delta w_n^{t-j-1}\|^2\big].$$

Proof. By the definition of e_n^t and the property of given compressor $\mathcal{C}(\cdot)$, we have

$$\begin{aligned}\mathbb{E}\big[\|e_n^t\|^2\big] &= \mathbb{E}\big[\|\varphi(\Delta w_n^{t-1}) + e_n^{t-1} - \mathcal{C}\big(\varphi(\Delta w_n^{t-1}) + e_n^{t-1}\big)\|^2\big]\\ &\leq (1-\rho)\mathbb{E}\big[\|\varphi(\Delta w_n^{t-1}) + e_n^{t-1}\|^2\big]\\ &\leq \left(\frac{\tau+1}{\tau}\right)(1-\rho)\mathbb{E}\big[\|\varphi(\Delta w_n^{t-1})\|^2\big] + (\tau+1)(1-\rho)\mathbb{E}\big[\|e_n^{t-1}\|^2\big].\end{aligned}$$

Here if we choose $\tau = \frac{\rho}{2(1-\rho)}$, then we obtain

$$\begin{aligned}\mathbb{E}\big[\|e_n^t\|^2\big] &\leq \frac{(2-\rho)(1-\rho)}{\rho}\mathbb{E}\big[\|\varphi(\Delta w_n^{t-1})\|^2\big] + \left(\frac{2-\rho}{2}\right)\mathbb{E}\big[\|e_n^{t-1}\|^2\big]\\ &\leq \frac{2(1-\rho)}{\rho}\mathbb{E}\big[\|\varphi(\Delta w_n^{t-1})\|^2\big] + \left(\frac{2-\rho}{2}\right)\mathbb{E}\big[\|e_n^{t-1}\|^2\big]\\ &= \frac{2(1-\rho)}{\rho}\mathbb{E}\big[\|\Delta w_n^{t-1}\|^2\big] + \left(\frac{2-\rho}{2}\right)\mathbb{E}\big[\|e_n^{t-1}\|^2\big]\end{aligned}$$

where the second inequality is due to the fact that $(2-\rho)(1-\rho) \leq 2(1-\rho)$ for $0 < \rho \leq 1$, and the last equality is due to Property A.4. Thus unfolding the above recursive equation with $e_n^0 = 0$ yields

$$\mathbb{E}\big[\|e_n^t\|^2\big] \leq \frac{2(1-\rho)}{\rho}\sum_{j=0}^{t-1}\left(\frac{2-\rho}{2}\right)^j\mathbb{E}\big[\|\Delta w_n^{t-j-1}\|^2\big]$$

Taking summation over $n = 1, \ldots, N$ on both sides results in a desired bound. ☐

Now we ready to prove the following convergence guarantee.

Theorem A.7. *Let $F_n(w)$ adhere to assumption A.1 and A.2, and suppose compressor $\mathcal{C}(\cdot)$ induces a compression error to the extent of at most $1-\rho$ in terms of l_2 norm, i.e., $\|g-\mathcal{C}(g)\|^2 \le (1-\rho)\|g\|^2$. Let $\sigma_l^2 = \max_{n\in[N]}\{\sigma_n^2\}$ and $C_1 = \frac{3}{4} + \frac{8(1-\rho)}{\rho^2}$. Then if we choose the learning rate that satisfies $\eta_t \le \frac{1}{\sqrt{144 C_1 H L}}$ in Algorithm 1, it holds that*

$$\frac{1}{\Omega_T}\sum_{t=0}^{T-1}\eta_t\mathbb{E}\big[\|\nabla F(w^t)\|^2\big] \lesssim \frac{F(w^0)-F(w^*)}{\Omega_T H} + \frac{L\sigma_l^2}{2\Omega_T N}\sum_{t=0}^{T-1}\eta_t^2 + \frac{3C_1 H L^2\big(6H\sigma_g^2+\sigma_l^2\big)}{\Omega_T}\sum_{t=0}^{T-1}\eta_t^3,$$

where $w^* = \arg\min_{w\in\mathbb{R}^d} F(w)$ and $\Omega_T = \sum_{t=0}^{T-1}\eta_t$.

Proof. Before delve into the detailed proof, we introduce the intermediate variable, $\widetilde{w}^t := w^t + \varphi^{-1}(\frac{1}{N}\sum_{n=1}^N e_n^t)$. Here we define $w^{t,i} = \frac{1}{N}\sum_{n=1}^N w_n^{t,i}$ for $i = 0,1,\dots,H$. Now let us unroll the $\frac{1}{N}\sum_{n=1}^N e_n^{t+1}$ by the definition as,

$$\frac{1}{N}\sum_{n=1}^N e_n^{t+1} = \frac{1}{N}\sum_{n=1}^N\big(\varphi(\Delta w_n^t)+e_n^t\big) - \frac{1}{N}\sum_{n=1}^N\mathcal{C}\big(\varphi(\Delta w_n^t)+e_n^t\big) \tag{9}$$

And we rewrite the update rule after aggregating the gradients recieved from the clients as follows.

$$w^{t+1} = w^t + \varphi^{-1}\left(\frac{1}{N}\sum_{n=1}^N\mathcal{C}\big(\varphi(\Delta w_n^t)+e_n^t\big)\right) \tag{10}$$

Then if we take inverse mapping $\varphi^{-1}(\cdot)$ on both sides of Eq 9, and add it to both sides of Eq 10, we have

$$\begin{aligned}
w^{t+1} + \varphi^{-1}\left(\frac{1}{N}\sum_{n=1}^N e_n^{t+1}\right) &= w^t + \varphi^{-1}\left(\frac{1}{N}\sum_{n=1}^N\mathcal{C}\big(\varphi(\Delta w_n^t)+e_n^t\big)\right)\\
&\quad + \varphi^{-1}\left(\frac{1}{N}\sum_{n=1}^N\big(\varphi(\Delta w_n^t)+e_n^t\big) - \frac{1}{N}\sum_{n=1}^N\mathcal{C}\big(\varphi(\Delta w_n^t)+e_n^t\big)\right)\\
&= w^t + \varphi^{-1}\left(\frac{1}{N}\sum_{n=1}^N\mathcal{C}\big(\varphi(\Delta w_n^t)+e_n^t\big)\right)\\
&\quad + \varphi^{-1}\left(\frac{1}{N}\sum_{n=1}^N\big(\varphi(\Delta w_n^t)+e_n^t\big)\right) - \varphi^{-1}\left(\frac{1}{N}\sum_{n=1}^N\mathcal{C}\big(\varphi(\Delta w_n^t)+e_n^t\big)\right)\\
&= w^t + \frac{1}{N}\sum_{n=1}^N\varphi^{-1}(\varphi(\Delta w_n^t)) + \varphi^{-1}\left(\frac{1}{N}\sum_{n=1}^N e_n^t\right)\\
&= w^t + \varphi^{-1}\left(\frac{1}{N}\sum_{n=1}^N e_n^t\right) + \frac{1}{N}\sum_{n=1}^N\Delta w_n^t
\end{aligned}$$

where the second and third equality is due to Property A.3 of mapping $\varphi(\cdot)$ and $\varphi^{-1}(\cdot)$.

Then by the definition of intermediate variable $\widetilde{w}^t$, for a specific round of t, the following holds,

$$\widetilde{w}^{t+1} - \widetilde{w}^t = \frac{1}{N}\sum_{n=1}^N\Delta w_n^t = -\frac{1}{N}\sum_{n=1}^N\sum_{i=0}^{H-1}\eta_t\widetilde{\nabla}F_n(w_n^{t,i}).$$

By L-smoothness property of $F(w) = \frac{1}{N}\sum_{n=1}^N F_n(w)$ and taking expectation on both sides, we have

$$\begin{aligned}
&\mathbb{E}[F(\widetilde{w}^{t+1}) - F(\widetilde{w}^t)] \le \mathbb{E}[\langle\nabla F(\widetilde{w}^t), \widetilde{w}^{t+1}-\widetilde{w}^t\rangle] + \frac{L}{2}\mathbb{E}[\|\widetilde{w}^{t+1}-\widetilde{w}^t\|^2]\\
&\overset{(a)}{=} \mathbb{E}\Big[\Big\langle\nabla F(\widetilde{w}^t), -\frac{1}{N}\sum_{n=1}^N\sum_{i=0}^{H-1}\eta_t\nabla F_n(w_n^{t,i})\Big\rangle\Big] + \frac{\eta_t^2 L}{2}\mathbb{E}\Big[\Big\|\frac{1}{N}\sum_{n=1}^N\sum_{i=0}^{H-1}\widetilde{\nabla}F_n(w_n^{t,i})\Big\|^2\Big]
\end{aligned}$$

$$
\begin{aligned}
&= \underbrace{\mathbb{E}\Big[\Big\langle \nabla F(w^t), -\frac{1}{N}\sum_{n=1}^{N}\sum_{i=0}^{H-1}\eta_t \nabla F_n(w_n^{t,i})\Big\rangle\Big]}_{A_1} + \eta_t \underbrace{\mathbb{E}\Big[\Big\langle \nabla F(w^t) - \nabla F(\widetilde{w}^t), \frac{1}{N}\sum_{n=1}^{N}\sum_{i=0}^{H-1}\nabla F_n(w_n^{t,i})\Big\rangle\Big]}_{A_2} \\
&\quad + \frac{\eta_t^2 L}{2}\underbrace{\mathbb{E}\Big[\Big\|\frac{1}{N}\sum_{n=1}^{N}\sum_{i=0}^{H-1}\widetilde{\nabla} F_n(w_n^{t,i})\Big\|^2\Big]}_{A_3}
\end{aligned}
\tag{11}
$$

where (a) comes from taking expectation with respect to stochasticity of the gradient due to mini-batch. Now we start with the term A_1 as follows,

$$
\begin{aligned}
A_1 &= \mathbb{E}\Big[\Big\langle \nabla F(w^t), -\frac{1}{N}\sum_{n=1}^{N}\sum_{i=0}^{H-1}\eta_t \nabla F_n(w_n^{t,i}) + \eta_t H \nabla F(w^t) - \eta_t H \nabla F(w^t)\Big\rangle\Big] \\
&= -\eta_t H \mathbb{E}\big[\|\nabla F(w^t)\|^2\big] + \underbrace{\mathbb{E}\Big[\Big\langle \nabla F(w^t), -\frac{1}{N}\sum_{n=1}^{N}\sum_{i=0}^{H-1}\eta_t \nabla F_n(w_n^{t,i}) + \eta_t H \nabla F(w^t)\Big\rangle\Big]}_{A_1^{'}}.
\end{aligned}
$$

And then we bound the term $A_1^{'}$ as,

$$
\begin{aligned}
A_1^{'} &= \mathbb{E}\Big[\Big\langle \nabla F(w^t), -\frac{1}{N}\sum_{n=1}^{N}\sum_{i=0}^{H-1}\eta_t \nabla F_n(w_n^{t,i}) + \eta_t H \nabla F(w^t)\Big\rangle\Big] \\
&= \mathbb{E}\Big[\Big\langle \nabla F(w^t), -\frac{1}{N}\sum_{n=1}^{N}\sum_{i=0}^{H-1}\eta_t \nabla F_n(w_n^{t,i}) + \frac{1}{N}\sum_{n=1}^{N}\sum_{i=0}^{H-1}\eta_t \nabla F_n(w^t)\Big\rangle\Big] \\
&= \mathbb{E}\Big[\Big\langle \sqrt{\eta_t H}\nabla F(w^t), -\frac{\sqrt{\eta_t}}{N\sqrt{H}}\sum_{n=1}^{N}\sum_{i=0}^{H-1}\Big(\nabla F_n(w_n^{t,i}) - \nabla F_n(w^t)\Big)\Big\rangle\Big] \\
&\overset{(c)}{=} \frac{\eta_t H}{2}\mathbb{E}\big[\|\nabla F(w^t)\|^2\big] - \frac{\eta_t}{2HN^2}\mathbb{E}\Big[\Big\|\sum_{n=1}^{N}\sum_{i=0}^{H-1}\nabla F_n(w_n^{t,i})\Big\|^2\Big] + \frac{\eta_t}{2HN^2}\mathbb{E}\Big[\Big\|\sum_{n=1}^{N}\sum_{i=0}^{H-1}\Big(\nabla F_n(w_n^{t,i}) - \nabla F_n(w^t)\Big)\Big\|^2\Big] \\
&\overset{(d)}{\le} \frac{\eta_t H}{2}\mathbb{E}\big[\|\nabla F(w^t)\|^2\big] - \frac{\eta_t}{2HN^2}\mathbb{E}\Big[\Big\|\sum_{n=1}^{N}\sum_{i=0}^{H-1}\nabla F_n(w_n^{t,i})\Big\|^2\Big] + \frac{\eta_t}{2N}\sum_{n=1}^{N}\sum_{i=0}^{H-1}\mathbb{E}\Big[\Big\|\nabla F_n(w_n^{t,i}) - \nabla F_n(w^t)\Big\|^2\Big] \\
&\overset{(e)}{\le} \frac{\eta_t H}{2}\mathbb{E}\big[\|\nabla F(w^t)\|^2\big] - \frac{\eta_t}{2HN^2}\mathbb{E}\Big[\Big\|\sum_{n=1}^{N}\sum_{i=0}^{H-1}\nabla F_n(w_n^{t,i})\Big\|^2\Big] + \frac{\eta_t L^2}{2N}\sum_{n=1}^{N}\sum_{i=0}^{H-1}\mathbb{E}\big[\|w_n^{t,i} - w^t\|^2\big]
\end{aligned}
$$

where (c) comes from $\|z_1 - z_2\|^2 = \|z_1\|^2 + \|z_2\|^2 - 2\langle z_1, z_2\rangle$, and (d) comes from Jensen's inequality, and (e) is due to L-smoothness of $F_n(w)$. Next, by the assumption A.2 and the fact that $\mathbb{E}[\|z - \mathbb{E}[z] + \mathbb{E}[z]\|^2] = \mathbb{E}[\|z - \mathbb{E}[z]\|^2] + \mathbb{E}[\|\mathbb{E}[z]\|^2]$, the term A_3 is bounded as,

$$
\begin{aligned}
A_3 &= \mathbb{E}\Big[\Big\|\frac{1}{N}\sum_{n=1}^{N}\sum_{i=0}^{H-1}\widetilde{\nabla} F_n(w_n^{t,i})\Big\|^2\Big] \\
&= \mathbb{E}\Big[\Big\|\frac{1}{N}\sum_{n=1}^{N}\sum_{i=0}^{H-1}\Big(\widetilde{\nabla} F_n(w_n^{t,i}) - \nabla F_n(w_n^{t,i})\Big)\Big\|^2\Big] + \mathbb{E}\Big[\Big\|\frac{1}{N}\sum_{n=1}^{N}\sum_{i=0}^{H-1}\nabla F_n(w_n^{t,i})\Big\|^2\Big] \\
&\overset{(f)}{=} \frac{1}{N^2}\sum_{n=1}^{N}\sum_{i=0}^{H-1}\mathbb{E}\Big[\Big\|\widetilde{\nabla} F_n(w_n^{t,i}) - \nabla F_n(w_n^{t,i})\Big\|^2\Big] + \frac{1}{N^2}\mathbb{E}\Big[\Big\|\sum_{n=1}^{N}\sum_{i=0}^{H-1}\nabla F_n(w_n^{t,i})\Big\|^2\Big] \\
&\overset{(g)}{\le} \frac{H}{N^2}\sum_{n=1}^{N}\sigma_n^2 + \frac{1}{N^2}\mathbb{E}\Big[\Big\|\sum_{n=1}^{N}\sum_{i=0}^{H-1}\nabla F_n(w_n^{t,i})\Big\|^2\Big]
\end{aligned}
$$

where (f) comes from the fact that if $z_1, z_2, \dots, z_n$ are independent and zero mean random variables, it holds that $\mathbb{E}[\|z_1 + z_2 + \cdots + z_n\|^2] = \mathbb{E}[\|z_1\|^2] + \mathbb{E}[\|z_2\|^2] + \cdots + \mathbb{E}[\|z_n\|^2]$, and (g) is due to the Assumption A.2. Thirdly, the term A_2 is bounded as the following.

$$\begin{aligned}
A_2 &= \mathbb{E}\Big[\Big\langle \nabla F(w^t) - \nabla F(\widetilde{w}^t), \frac{1}{N}\sum_{n=1}^{N}\sum_{i=0}^{H-1} \nabla F_n(w_n^{t,i}) \Big\rangle\Big] \\
&\overset{(h)}{\le} \frac{1}{2a}\mathbb{E}\Big[\Big\|\nabla F(w^t) - \nabla F(\widetilde{w}^t)\Big\|^2\Big] + \frac{a}{2}\mathbb{E}\Big[\Big\|\frac{1}{N}\sum_{n=1}^{N}\sum_{i=0}^{H-1} \nabla F_n(w_n^{t,i})\Big\|^2\Big] \\
&\le \frac{L^2}{2a}\mathbb{E}[\|w^t - \widetilde{w}^t\|^2] + \frac{a}{2}\mathbb{E}\Big[\Big\|\frac{1}{N}\sum_{n=1}^{N}\sum_{i=0}^{H-1} \nabla F_n(w_n^{t,i})\Big\|^2\Big] \\
&\overset{(i)}{\le} 2HL^2\mathbb{E}[\|w^t - \widetilde{w}^t\|^2] + \frac{1}{8H}\mathbb{E}\Big[\Big\|\frac{1}{N}\sum_{n=1}^{N}\sum_{i=0}^{H-1} \nabla F_n(w_n^{t,i})\Big\|^2\Big] \\
&= 2HL^2\mathbb{E}[\|\varphi^{-1}(\frac{1}{N}\sum_{n=1}^{N} e_n^t)\|^2] + \frac{1}{8H}\mathbb{E}\Big[\Big\|\frac{1}{N}\sum_{n=1}^{N}\sum_{i=0}^{H-1} \nabla F_n(w_n^{t,i})\Big\|^2\Big] \\
&\overset{(j)}{\le} 2HL^2\frac{1}{N}\sum_{n=1}^{N}\mathbb{E}[\|e_n^t\|^2] + \frac{1}{8H}\underbrace{\mathbb{E}\Big[\Big\|\frac{1}{N}\sum_{n=1}^{N}\sum_{i=0}^{H-1} \nabla F_n(w_n^{t,i})\Big\|^2\Big]}_{A_4}
\end{aligned}$$

where (h) comes from the fact that $\langle z_1, z_2\rangle \le \frac{1}{2a}\|z_1\|^2 + \frac{a}{2}\|z_2\|^2$ for any $a > 0$, and (i) is derived by choosing $a = \frac{1}{4H}$, and (j) is due to Property A.4 and Jensen's inequality. Finally, we rewrite the term A_4 as shown below.

$$\begin{aligned}
A_4 &= \mathbb{E}\Big[\Big\|\frac{1}{N}\sum_{n=1}^{N}\sum_{i=0}^{H-1} \nabla F_n(w_n^{t,i})\Big\|^2\Big] \\
&= \mathbb{E}\Big[\Big\|\frac{1}{N}\sum_{n=1}^{N}\sum_{i=0}^{H-1} \nabla F_n(w_n^{t,i}) - \frac{1}{N}\sum_{n=1}^{N}\sum_{i=0}^{H-1} \nabla F_n(w^t) + \frac{1}{N}\sum_{n=1}^{N}\sum_{i=0}^{H-1} \nabla F_n(w^t)\Big\|^2\Big] \\
&\le 2\mathbb{E}\Big[\Big\|\frac{1}{N}\sum_{n=1}^{N}\sum_{i=0}^{H-1} \nabla F_n(w_n^{t,i}) - \frac{1}{N}\sum_{n=1}^{N}\sum_{i=0}^{H-1} \nabla F_n(w^t)\Big\|^2\Big] + 2\mathbb{E}\Big[\Big\|\frac{1}{N}\sum_{n=1}^{N}\sum_{i=0}^{H-1} \nabla F_n(w^t)\Big\|^2\Big] \\
&\le \frac{2H}{N}\sum_{n=1}^{N}\sum_{i=0}^{H-1}\mathbb{E}\Big[\Big\|\nabla F_n(w_n^{t,i}) - \nabla F_n(w^t)\Big\|^2\Big] + 2H^2\mathbb{E}\Big[\Big\|\underbrace{\frac{1}{N}\sum_{n=1}^{N}\nabla F_n(w^t)}_{=\nabla F(w^t)}\Big\|^2\Big] \\
&\le \frac{2HL^2}{N}\sum_{n=1}^{N}\sum_{i=0}^{H-1}\mathbb{E}[\|w_n^{t,i} - w^t\|^2] + 2H^2\mathbb{E}\Big[\|\nabla F(w^t)\|^2\Big]
\end{aligned}$$

where the second and third inequalities are due to Jensen's inequality and the last inequality comes from Assumption A.1. By putting it altogether, Eq (11) can be rewritten as follows.

$$\begin{aligned}
&\mathbb{E}[F(\widetilde{w}^{t+1}) - F(\widetilde{w}^t)] \le -\eta_t H\mathbb{E}[\|\nabla F(w^t)\|^2] + A_1' + \eta_t A_2 + \frac{\eta_t^2 L}{2}A_3 \\
&\le -\frac{\eta_t H}{2}\mathbb{E}[\|\nabla F(w^t)\|^2] - \frac{\eta_t}{2HN^2}\mathbb{E}\Big[\Big\|\sum_{n=1}^{N}\sum_{i=0}^{H-1}\nabla F_n(w_n^{t,i})\Big\|^2\Big] + \frac{\eta_t L^2}{2N}\sum_{n=1}^{N}\sum_{i=0}^{H-1}\mathbb{E}[\|w_n^{t,i} - w^t\|^2] \\
&\quad + 2\eta_t HL^2\frac{1}{N}\sum_{n=1}^{N}\mathbb{E}[\|e_n^t\|^2] + \frac{\eta_t L^2}{4N}\sum_{n=1}^{N}\sum_{i=0}^{H-1}\mathbb{E}[\|w_n^{t,i} - w^t\|^2] + \frac{\eta_t H}{4}\mathbb{E}[\|\nabla F(w^t)\|^2] \\
&\quad + \frac{\eta_t^2 HL}{2N^2}\sum_{n=1}^{N}\sigma_n^2 + \frac{\eta_t^2 L}{2N^2}\mathbb{E}\Big[\Big\|\sum_{n=1}^{N}\sum_{i=0}^{H-1}\nabla F_n(w_n^{t,i})\Big\|^2\Big]
\end{aligned}$$

$$
\begin{aligned}
&= -\frac{\eta_t H}{4}\mathbb{E}\big[\|\nabla F(w^t)\|^2\big] + \frac{3\eta_t L^2}{4N}\sum_{n=1}^{N}\sum_{i=0}^{H-1}\mathbb{E}\big[\|w_n^{t,i} - w^t\|^2\big] + 2\eta_t H L^2 \frac{1}{N}\sum_{n=1}^{N}\mathbb{E}\big[\|e_n^t\|^2\big] \\
&\quad + \underbrace{\left(\frac{\eta_t^2 L}{2N^2} - \frac{\eta_t}{2HN^2}\right)}_{\leq 0 \text{ when } \eta_t \leq \frac{1}{HL}}\mathbb{E}\Big[\Big\|\sum_{n=1}^{N}\sum_{i=0}^{H-1}\nabla F_n(w_n^{t,i})\Big\|^2\Big] + \frac{\eta_t^2 HL}{2N^2}\sum_{n=1}^{N}\sigma_n^2 \\
&\leq -\frac{\eta_t H}{4}\mathbb{E}\big[\|\nabla F(w^t)\|^2\big] + \frac{3\eta_t L^2}{4N}\sum_{n=1}^{N}\sum_{i=0}^{H-1}\mathbb{E}\big[\|w_n^{t,i} - w^t\|^2\big] \\
&\quad + \mathbb{1}_{[t\geq 1]}\cdot 2\eta_t H L^2 \frac{2(1-\rho)}{\rho}\frac{1}{N}\sum_{n=1}^{N}\sum_{j=0}^{t-1}\left(\frac{2-\rho}{2}\right)^j \mathbb{E}\big[\|\Delta w_n^{t-j-1}\|^2\big] + \frac{\eta_t^2 HL}{2N^2}\sum_{n=1}^{N}\sigma_n^2
\end{aligned}
$$

where the last inequality comes from Lemma A.6 and $\mathbb{1}_{[\cdot]}$ is the indicator function.

Then taking telescoping sum over $t = 0$ to $t = T-1$ on both sides yields,

$$
\begin{aligned}
\mathbb{E}[F(\widetilde{w}^T)] - F(w^0) &\leq -\frac{H}{4}\sum_{t=0}^{T-1}\eta_t\mathbb{E}\big[\|\nabla F(w^t)\|^2\big] + \frac{3L^2}{4N}\sum_{t=0}^{T-1}\eta_t\sum_{n=1}^{N}\sum_{i=0}^{H-1}\mathbb{E}\big[\|w_n^{t,i} - w^t\|^2\big] \\
&\quad + 2HL^2\frac{2(1-\rho)}{\rho}\frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T-1}\sum_{j=0}^{t-1}\left(\frac{2-\rho}{2}\right)^j \eta_t\mathbb{E}\big[\|\Delta w_n^{t-j-1}\|^2\big] + \frac{HL}{2N^2}\sum_{t=0}^{T-1}\eta_t^2\sum_{n=1}^{N}\sigma_n^2 \\
&\leq -\frac{H}{4}\sum_{t=0}^{T-1}\eta_t\mathbb{E}\big[\|\nabla F(w^t)\|^2\big] + \frac{3L^2}{4N}\sum_{t=0}^{T-1}\eta_t\sum_{n=1}^{N}\sum_{i=0}^{H-1}\mathbb{E}\big[\|w_n^{t,i} - w^t\|^2\big] \\
&\quad + 2HL^2\frac{2(1-\rho)}{\rho}\frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T-1}\sum_{j=0}^{T-2}\left(\frac{2-\rho}{2}\right)^j \eta_t\mathbb{E}\big[\|\Delta w_n^{t-1}\|^2\big] + \frac{HL}{2N^2}\sum_{t=0}^{T-1}\eta_t^2\sum_{n=1}^{N}\sigma_n^2 \\
&\leq -\frac{H}{4}\sum_{t=0}^{T-1}\eta_t\mathbb{E}\big[\|\nabla F(w^t)\|^2\big] + \frac{3L^2}{4N}\sum_{t=0}^{T-1}\eta_t\sum_{n=1}^{N}\sum_{i=0}^{H-1}\mathbb{E}\big[\|w_n^{t,i} - w^t\|^2\big] \\
&\quad + 2HL^2\frac{2(1-\rho)}{\rho}\frac{1}{N}\sum_{n=1}^{N}\sum_{j=0}^{\infty}\left(\frac{2-\rho}{2}\right)^j \sum_{t=0}^{T-1}\eta_t\mathbb{E}\big[\|\Delta w_n^{t}\|^2\big] + \frac{HL}{2N^2}\sum_{t=0}^{T-1}\eta_t^2\sum_{n=1}^{N}\sigma_n^2 \\
&\leq -\frac{H}{4}\sum_{t=0}^{T-1}\eta_t\mathbb{E}\big[\|\nabla F(w^t)\|^2\big] + \frac{3L^2}{4N}\sum_{t=0}^{T-1}\eta_t\sum_{n=1}^{N}\sum_{i=0}^{H-1}\mathbb{E}\big[\|w_n^{t,i} - w^t\|^2\big] \\
&\quad + 2HL^2\frac{4(1-\rho)}{\rho^2}\sum_{t=0}^{T-1}\eta_t\frac{1}{N}\sum_{n=1}^{N}\mathbb{E}\big[\|w_n^{t,H} - w^t\|^2\big] + \frac{HL}{2N^2}\sum_{t=0}^{T-1}\eta_t^2\sum_{n=1}^{N}\sigma_n^2 \\
&\leq -\frac{H}{4}\sum_{t=0}^{T-1}\eta_t\mathbb{E}\big[\|\nabla F(w^t)\|^2\big] + \frac{3HL^2}{4}\sum_{t=0}^{T-1}\eta_t\left[3H\eta_t^2\left(6H\sigma_g^2 + \frac{1}{N}\sum_{n=1}^{N}\sigma_n^2\right) + 18H^2\eta_t^2\mathbb{E}\big[\|\nabla F(w^t)\|^2\big]\right] \\
&\quad + \frac{8(1-\rho)HL^2}{\rho^2}\sum_{t=0}^{T-1}\eta_t\left[3H\eta_t^2\left(6H\sigma_g^2 + \frac{1}{N}\sum_{n=1}^{N}\sigma_n^2\right) + 18H^2\eta_t^2\mathbb{E}\big[\|\nabla F(w^t)\|^2\big]\right] + \frac{HL}{2N^2}\sum_{t=0}^{T-1}\eta_t^2\sum_{n=1}^{N}\sigma_n^2 \\
&= -\frac{H}{4}\sum_{t=0}^{T-1}\eta_t\mathbb{E}\big[\|\nabla F(w^t)\|^2\big] + \frac{HL}{2N^2}\sum_{t=0}^{T-1}\eta_t^2\sum_{n=1}^{N}\sigma_n^2 \\
&\quad + \left(\frac{3HL^2}{4} + \frac{8(1-\rho)HL^2}{\rho^2}\right)\sum_{t=0}^{T-1}\eta_t\left[3H\eta_t^2\left(6H\sigma_g^2 + \frac{1}{N}\sum_{n=1}^{N}\sigma_n^2\right) + 18H^2\eta_t^2\mathbb{E}\big[\|\nabla F(w^t)\|^2\big]\right]
\end{aligned}
$$

where the second and third inequalities hold when η_t is non-increasing, which falls within our scope of interest, and the second last inequality comes from Lemma A.5. If we define $\sigma_l^2 := \max_{n\in[N]}\{\sigma_n^2\}$ and $C_1 := \frac{3}{4} + \frac{8(1-\rho)}{\rho^2}$, then we can simplify the above derivation as follows.

$$\mathbb{E}[F(\widetilde{w}^T)] - F(w^0) \leq \sum_{t=0}^{T-1} \underbrace{\left(-\frac{\eta_t H}{4} + 18C_1\eta_t^3 H^3 L^2 \right)}_{A_5} \mathbb{E}\big[\|\nabla F(w^t)\|^2\big]$$
$$+ \sum_{t=0}^{T-1} 3C_1\eta_t^3 H^2 L^2 \big(6H\sigma_g^2 + \sigma_l^2\big) + \sum_{t=0}^{T-1} \frac{\eta_t^2 HL}{2N}\sigma_l^2$$

Here if we choose the learning rate that satisfies $\eta_t \leq \frac{1}{\sqrt{144C_1HL}}$, the term A_5 is bounded as $A_5 \leq -\frac{\eta_t H}{8}$. Finally, rearranging the terms and dividing $\Omega_T := \sum_{t=0}^{T-1} \eta_t$ on both sides yields the following results.

$$\frac{H}{8\Omega_T} \sum_{t=0}^{T-1} \eta_t \mathbb{E}\big[\|\nabla F(w^t)\|^2\big] \leq \frac{F(w^0) - F^*}{\Omega_T} + \frac{HL}{2\Omega_T N}\sigma_l^2 \sum_{t=0}^{T-1} \eta_t^2 + \frac{3C_1H^2L^2\big(6H\sigma_g^2 + \sigma_l^2\big)}{\Omega_T} \sum_{t=0}^{T-1} \eta_t^3$$

where F^* is the global minimum value of $F(w)$, which completes the proof.

□

Corollary A.8. *Under conditions stated in Theorem A.7, for a fixed learning rate $\eta_t = \eta$ for all t, that is chosen to satisfy $\eta = \Theta\Big(\frac{\sqrt{N}}{\sqrt{TH}}\Big)$, the global model trained according to Algorithm 1 exhibits the following convergence behavior.*

$$\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\big[\|\nabla F(w^t)\|^2\big] = \mathcal{O}\bigg(\frac{F(w^0) - F(w^*)}{\sqrt{NTH}} + \frac{L\sigma_l^2}{\sqrt{NTH}} + \frac{NC_1L^2}{T}\big(6H\sigma_g^2 + \sigma_l^2\big)\bigg).$$

That is, our convergence result achieves a linear speedup with respect to the number of clients N, i.e., achieves a rate of $\mathcal{O}\Big(\frac{1}{\sqrt{NTH}}\Big)$ as long as $H = \mathcal{O}\Big(\frac{T^{1/3}}{N}\Big)$.

A.2. Design Parameters

Throughout all the simulations, we use SGD optimizer with learning rate 0.1, which is decayed with a rate of 0.998 at each round. The weight decay is set to be 5e-4. Following the implementation of current works (Qu et al., 2022b; Gao et al., 2022; Acar & Saligrama, 2022; Acar et al., 2021), the local gradient norm clipping is adopted. Moreover, the local epoch is set to be 5 with batch size of 50, and we run FL training for 600 rounds in total. Regarding the FedPara (Hyeon-Woo et al., 2022), the compression ratio is set to be 0.2 for each layer. We conduct all the experiments for 3 times with different random seeds and report the averaged values.

Note that FL training at the initial stage may drastically change the model parameters. Thus we set the period for constructing and broadcasting of the mapping function $\mathcal{P}$ as follows: the server constructs and broadcasts the mapping every 20 round at the first 100 rounds, every 50 round during rounds 100 to 300, and every 100 round for the remaining rounds from 300 to 600. We conduct the ablation on how varying period for constructing/broadcasting affects the performance in Sec. A.8.

A.3. Sparsifying Attribute of the Mapping for the Case of Multiple Iterations

Here we describe our sparsifying attribute of the mapping approach in case of multiple iterations induced from multiple local updates H and mini-batch SGD in detail. Recall that we define the local gradient as the weight difference between the parameters before and after the local update, i.e, $\Delta w_n = w_n - w$ with multiple number of iterations H, as we mentioned in the main manuscript. However, as explained in Eq (1), for the sake of easier understanding, we presented the formula in its simplest form, considering a single batch and iteration. There appears to be a slight discrepancy between the explanation and actual operation, due to the need to account for the number of iterations.

Here we would like to emphasize that our lossless sparsification property of our mapping still holds, even if we account for multiple number of iterations in the gradient computation. If we consider multiple local iterations, Eq (1) in main paper can be rewritten in the general form as follows: $dW = \sum_{i=0}^{H-1} dz^i \cdot a^{i\top}$ where dz^i and $a^{i\top}$ denote the input and output gradient computed at i-th iteration respectively. Since the mapping operation is linear, the following holds:

$$\varphi(dW) = Q^\top dWP = Q^\top(\sum_{i=0}^{H-1} dz^i \cdot a^{i\top})P = \sum_{i=0}^{H-1}(Q^\top dz^i \cdot a^{i\top} P).$$

In other words, taking the mapping operation after the summation is equivalent to the summation after mapping. Here, the gradient $Q^\top dz^i \cdot a^{i\top} P$ for each iteration i is obviously the sparsified gradient, i.e., $\varphi(dW)$ is the summation of the sparsified gradients. Since the gradient computed at each iteration $dz^i \cdot a^{i\top}$ is computed from the local data following the same local data distribution within a single client, the positions of elements tending towards near-zero would be similar. Hence, the summation of each sparsified gradient, $\sum_{i=0}^{H-1}(Q^\top dz^i \cdot a^{i\top} P)$, becomes sparsified more than $dW = \sum_{i=0}^{H-1} dz^i \cdot a^{i\top}$, which is also supported by the fact that specifying the local epochs as 5 (the total number of iterations becomes about 50 in CIFAR10 and CIFAR100) still successfully improves the communication efficiency compared to the baselines, as demonstrated in the main experimental results.

Moreover, the gradient can indeed be expressed as $\frac{1}{B}\sum_{i=1}^{B} dz^{(i)} \cdot a^{(i)\top}$ in case of mini-batch SGD, where we again define $dz^{(i)}$ and $a^{(i)}$ to be input and output gradient vectors of i-th data sample. As the gradient can be written in the form of linear combination of gradient computed on each i-th sample, the same argument can be applied here, i.e., mapping operation after the summation is equivalent to the summation after the mapping. Therefore, the gradient computed with mini-batch SGD also becomes sparsified gradient in the mapping space, and this is directly evidenced by Figure 1b in the main manuscript, which is the very plot that shows the magnitude of gradients computed with batch size of 50.

A.4. Detailed Mechanism of the Mapping

Regarding more detailed description of our method, we again define A_l and Z_l with concrete dimension as follows. We stack the vectors of input (output gradient) computed on each i-th data samples in A_l (Z_l), where the batch information is already incorporated in these matrices, as follows:

$$A_l = [a_l^{(1)}, a_l^{(2)}, \ldots, a_l^{(B)}] \in \mathbb{R}^{m\times B},$$

$$Z_l = [dz_l^{(1)}, dz_l^{(2)}, \ldots, dz_l^{(B)}] \in \mathbb{R}^{n\times B}$$

where $a_l^{(i)} \in \mathbb{R}^m$ and $dz_l^{(i)} \in \mathbb{R}^n$. Thus we can directly exploit the structures, by considering each input and output gradient computed on all the samples one by one. Moreover, the mapping computation $\varphi(\Delta w)$ can be rewritten as shown in Eq (5):

$$\varphi(\Delta w) = [\varphi_1(dW_1); \ldots; \varphi_L(dW_L)]$$

where $\varphi_l(dW_l) = Q_l^\top dW_l P_l$, i.e., apply the mapping φ_l, which is defined with Q_l, P_l derived at each layer, to the gradient of each weight matrix W_l. In addition, when we apply the compression operator to the mapped gradient as $\mathcal{C}(\varphi(\Delta w))$, The operator can also be seen as being applied to each sparsified gradient, where each dW_l is mapped to $\varphi_l(dW_l)$ for all layers $l = 1, \ldots, L$, and then stacked together, i.e., $\mathcal{C}(\varphi(\Delta w)) = [\mathcal{C}(\varphi_1(dW_1)); \ldots; \mathcal{C}(\varphi_L(dW_L))]$.

A.5. Details on Baselines Utilizing Public Data

Since our mapping function necessitates the use of public data for the server, we also consider two more baselines that utilize the public data to make our comparisons as fair as possible. Specifically for these baselines, we relieve the assumption so that the clients can access the public data as well. Each client is then able to upload its own predicted output probability on the shared public data to the server. Then the server can aggregate (ensemble) the output probabilities that have been received from participating clients in every communication round. There are two possibilities to integrate the public data into FL training as shown below.

Approach 1. ***Directly combining server-side gradient (PD1)***

The first approach directly combines the server-side gradient into the aggregated gradient. The server-side gradient is computed from the distillation loss with the ensembled output probability on the shared public data. Specifically, we modify the aggregation rule in Eq. (7) as follows,

$$\Delta w^t = \frac{1}{N+1}\left(\sum_{n=1}^{N} \varphi^{-1}\Big(\mathcal{C}(\Delta_\varphi w_n^t)\Big) + \Delta w_{\text{server}}^t\right) \tag{12}$$

where $\Delta w^t_{\text{server}}$ refers to the amount of local update, starting from global model of t-th round w^t, with distillation loss computed on the server. In words, the server is treated as one of the participating nodes and we directly incorporate the server-side gradient into aggregated gradient by averaging altogether. Note that the predicted output probability for the public data on each client is obtained using its locally trained model $w_n^{t,H}$ that has been updated through the local iterations of H for all $n = 1, \ldots, N$. After computing the global update as in Eq. (12), the server broadcasts it to all the participating clients.

Approach 2. ***Distilling after the model update (PD2)***

The second approach for utilizing the public data is to train the model on the server-side right after updating the model with the aggregated gradient. Specifically, upon receiving the predicted output probabilities and the local updates, the server update the global model using the aggregated gradient in Eq. (7) to get the intermediate model as follows,

$$w^{t+\frac{1}{2}} \leftarrow w^t + \frac{1}{N}\sum_{n=1}^{N} \varphi^{-1}\big(\mathcal{C}(\Delta_\varphi w_n^t)\big).$$

Starting from $w^{t+\frac{1}{2}}$, the server then trains the model on the shared public data via distillation loss with ensembled probability of the public data. To put it clearly, the server initializes the model as $w^{t+\frac{1}{2},0} \leftarrow w^{t+\frac{1}{2}}$ and conducts local update with the following update rule.

$$w^{t+\frac{1}{2},i+1} = w^{t+\frac{1}{2},i} - \eta_t \widetilde{\nabla} F_{\text{server}}(w^{t+\frac{1}{2},i}) \quad \text{for} \quad i = 0, \ldots, H-1$$

where $\widetilde{\nabla} F_{\text{server}}(w^{t+\frac{1}{2},i})$ is the unbiased estimator of the true gradient computed on the server-side via distillation loss. After finishing the server-side local update, the server computes the total amount of global update as $\Delta w^t = w^{t+\frac{1}{2},H} - w^t$ and then broadcasts it to all the clients.

In both approach 1 and approach 2, upon receiving the global update Δw^t on the client-side, each client updates the model as $w^{t+1} \leftarrow w^t + \Delta w^t$, which synchronizes the global model. This process repeats until the global model converges.

A.6. Extra Costs For Employing Mapping Approach in FL

Although we focus on reducing uplink transmission which is the main bottleneck in FL, there are two additional costs when employing our mapping approach; 1) server-side computation for mapping computation, and 2) downlink communication load for broadcasting mapping function.

Table 3: Elapsed time (sec) for mapping construction on the server-side and local training per client per round on the client-side.

	Mapping construction	**Local training**
CIFAR10 (CNN)	3.5906	1.3708
CIFAR100 (ResNet18)	47.9614	3.6108

Computation cost on the server-side. Since our mapping construction necessitates the process of singular value decomposition (SVD), of which the computational complexity is known as $\mathcal{O}(n^3)$ given $n \times n$ matrix, it appears that our mapping construction introduces significant latency into FL training rounds. However the server typically has sufficient computation power, which allows for relatively fast computation of SVD. In addition, since we can adopt relatively longer period (cycle) for mapping broadcasting as FL rounds progresses as we set in Sec. A.2, the computation burden decreases in later stage of FL. Table 3 provides the elapsed time for constructing mapping on CIFAR10 (CNN) and CIFAR100 (ResNet18) along with the real time taken for local training per each client per round. As can be seen, the time taken to construct mapping function is not that significant compared to that of the local iterations, which are conducted hundreds of times during FL training. Moreover, as discussed in Sec. A.8, we can see that a longer period of mapping construction to reduce the computational burden is still effective in communication-efficient FL.

Downlink cost for mapping broadcasting. Regarding the communication load, since the global model can be broadcast to all clients using sufficient bandwidth during downlink communication, reducing the uplink transmission is more significant. However, there is an extra downlink cost for broadcasting of the mapping function at a certain round of $t \in \mathcal{P}$ relative to the baselines. We provide the experimental result of our approach that incorporates the additional bits communicated during downlink communication (for mapping broadcasting), denoted by 'kSB + Map (with downlink)' as shown in Figure 7. All other methods without '(with downlink)' only consider uplink transmission. Although our approach incorporating downlink bits for mapping falls slightly short compared to the baselines in terms of communication efficiency, it still demonstrates higher efficiency compared to FedAvg (which considers the uplink only), and still provides the benefit in terms of accuracy compared to all the baselines. In addition, we want to emphasize that as per the size of communication load incurred in FedAvg during the uplink, the downlink cost of mapping may be negligible compared to other baselines when incorporating the communication load for broacasting the aggregated gradient in every round across all the baselines.

Moreover, our strategy can be seen as the new space offers an advantage of offloading the heavy communication burdens from the client-side to the server-side. This is a favorable trade-off since the downlink tends to have a higher communication rate (allowable rate at which the data or information is transmitted) compared to the uplink. Moreover, as outlined in the case of computational cost, we can see that a longer period of mapping construction to reduce the downlink communication is still valid as shown in Sec. A.8.

Finally, there could be a possible solution to mitigate this limitation. If we relieve the assumption that the public data can also be available at the client-side as done in PD baselines, the mapping can be constructed by the client itself without the need for broadcasting of the mapping from the server. Moreover, a sufficiently effective mapping can be constructed even with access to a very small amount of data, which is already validated in Sec. 6.3.

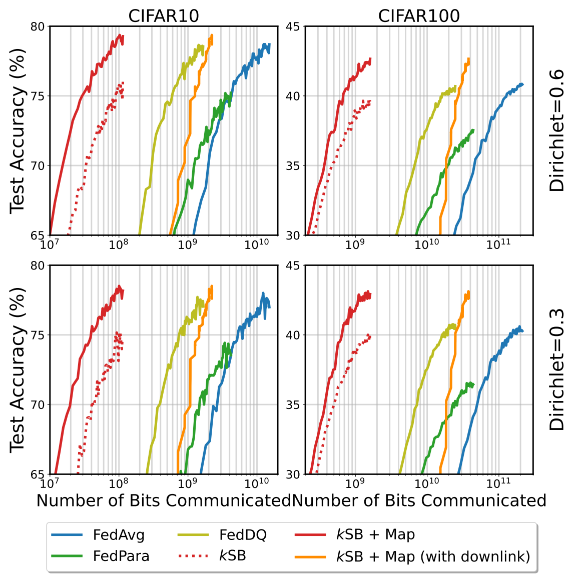


Figure 7: Accuracy versus number of communication bits on CIFAR10 and CIFAR100 under Dirichlet distribution with $\alpha = 0.6$ **(top row)** and $\alpha = 0.3$ **(bottom row)** among clients ($s = 0.03$ for kSB and kSB + Map).

A.7. Practical Implementation

For the ease of implementing our mapping process, we transfer the weight matrix instead of the gradient in practice, which allows for the gradient to be automatically transferred to the mapped space during backward propagation. This efficiency is thanks to the equivalence of the mapped gradient of the weight parameter and the gradient of the mapped weight parameter. Indeed, this implementation keeps the original algorithmic behavior exactly the same as it is. We adopt the implementation from (Kim et al., 2023) where the new weight space is explored in the context of continual learning, which turns out to be interestingly similar to our mapping process. To explain the equivalence of the gradient of the mapped weight parameter and the mapped gradient of the original weight parameter, we first express the weight matrix W_{orig} for a specific layer as follows.

$$W_{\text{orig}} = QW_{\text{new}}P^\top \tag{13}$$

where Q and P are orthogonal matrices obtained from Eq. (2). Then W_{new} can be written as,

$$W_{\text{new}} = Q^\top W_{\text{orig}}P = \varphi(W_{\text{orig}}). \tag{14}$$

On the one hand, by the chain rule, one can easily verify that the gradient of W_{new} can be derived as follows.

$$dW_{\text{new}} = Q^\top dW_{\text{orig}}P = \varphi(dW_{\text{orig}}) \tag{15}$$

Thus, once we transfer the weight as in Eq. (14) and reparameterize the W_{orig} as in Eq. (13), then we can automatically map the gradient during gradient computation through autograd, provided by well-known machine learning framework such as Pytorch. Then whenever the mapping function is newly updated and broadcast, all the clients and the server convert the current space which the weight matrices lie within into the new space as the following sense.

$$W_{\text{new}} \leftarrow \varphi_{\text{curr}}(\varphi_{\text{prev}}^{-1}(W_{\text{new}})) \tag{16}$$

where $\varphi_{\text{curr}}(\cdot)$, $\varphi_{\text{prev}}(\cdot)$ denote the new and previous mapping function respectively. Then the weight matrices are again reparameterized as in Eq. (13) and repeat the FL process. After the server broadcasts the aggregated gradient to all the clients, i.e., broadcast $\Delta w_{\text{new}} := \sum_{n=1}^{N} \kappa_n \mathcal{C}(\Delta_\varphi w_n^t)$, then each local device updates its model without transferring the aggregated gradient to the original space as shown below.

$$w_{\text{new}}^{t+1} = w_{\text{new}}^t + \Delta w_{\text{new}}^t \tag{17}$$

Table 4: Elapsed time (sec) for gradient computation and gradient + mapping process per one local iteration on each dataset.

	CIFAR10	**CIFAR100**	**TinyImageNet**
Gradient	0.00261	0.01901	0.02003
Gradient + Mapping	0.00428	0.02890	0.03750

where w_{new}^t refers to the weight of t-th global model, which comprises newly parameterized weights W_{new} for all layers.

Based on this implementation, we measured the elapsed time for mapping process on various datasets, including CIFAR10 for CNN (consists of two 5×5 convolution layers followed by three fully-connected layers), CIFAR100 and TinyImageNet for ResNet18. Specifically, we measured the time required for gradient computation and the time encompassing both gradient computation and mapping process, per one local iteration in seconds. As can be seen in Table 4, the elapsed time for encompassing both gradient computation and mapping (referred to as 'Gradient + Mapping') is less than twice than the duration time required for gradient computation only (referred to as 'Gradient'). In other words, the time taken for mapping process is less than the elapsed time for gradient computation. Although the time duration for the mapping process is not negligible, it can be considered a favorable trade-off for the improvements in both communication efficiency and performance.

A.8. Effect of Period of Mapping Construction

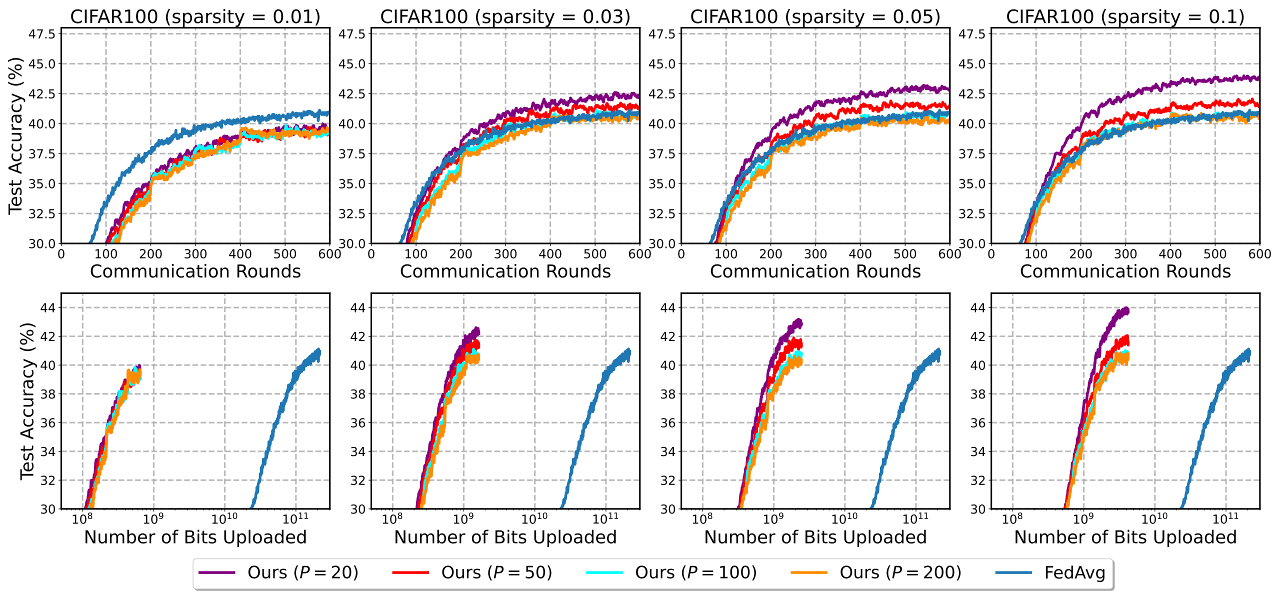


Figure 8: The effects of period of mapping construction on the performance and communication efficiency under $\alpha = 0.6$ on CIFAR100.

In this section, we conduct the ablation on varying period of construction of mapping function across various sparsity level, i.e., sparsity $= \{0.01, 0.03, 0.05, 0.1\}$, for kSB applied in our mapping (referred to as 'Ours' in the Figure 8) on CIFAR100. We evaluate our approach for periods of 20, 50, 100 and 200, where $P = 20$ in Figure 8 indicates the construction and broadcasting of mapping function by the server every 20 communication rounds. As can be seen in the top row, which represents the accuracy versus communication rounds, it is likely to observe the higher performance in shorter periods setup as sparsity increases. Although performance of our approach falls short of FedAvg when sparsity level is 0.01, it becomes comparable to or even outperforms FedAvg from 0.03 sparsity level. Thus, as can be seen in the bottom row of the figure, which represents the accuracy versus the number of bits, good communication efficiency can be guaranteed with large P. This indicates that the favorable results can be achieved without the need for frequent mapping construction.

A.9. Sparse Encoding Strategy

In measuring the communication bits required to upload a sparse gradient vector $g \in \mathbb{R}^d$, we must consider not only the bit-width for representing each non-zero element but also its position information. Here we adopt block sparse encoding from (Ozfatura et al., 2021) when estimating the position bits during the uplink transmission. To encode the position of the non-zero element in g, we first partition the vector into multiple disjoint blocks of equal size. If each block is of size d_B, then $\log_2(d_B)$ bits are sufficient to fully encode/decode every positions of the elements that we desire to upload within each block. Now we start from the first block to store the location of the non-zero elements. If there is non-zero element in the current block, store the position using $\log_2(d_B) + 1$ bits. The additional 1 bit used here is to indicate whether or not there is non-zero element. For example, when the current non-zero element is 5-th position (position counting starts from 0-th) within the current block of size $d_B = 16$, the encoding bits would be

$$\underbrace{1}_{\text{Indicator}} \ \underbrace{0110}_{\text{Position}} . \tag{18}$$

If there is no more non-zero elements in the current block, insert a 0 into the encoding stream to indicate the corresponding status and then go to the next block. This encoding process repeats until there is no more non-zero elements in the last block. Given this encoded bits stream, the decoding strategy is as follows. **a**) Initialize the current block as the first block. **b**) Check whether the first bit of encoded stream is 1 or 0. *If this indicator bit is 1*, read the next $\log_2(d_B)$ bits to decode the position within the current block. **c**) Read the next bit and return to the step **b**). *If the indicator bit is 0*, go to the next block, set it as the current block and return to the step **c**). This process repeats until the last bit is encountered.

Given the above strategy, the position bits for each non-zero elements would be $\log_2(d_B) + 1$ (including the indicator bit for 1), and the indicator bit for 0 would show up $\lceil d/d_B \rceil$, i.e., total number of blocks, times in total. Following the (Ozfatura et al., 2021), we set the size of each block to be $d_B = 2^{\lfloor \log_2(1/s) \rfloor}$. Hence, the required position bits in total would be

$$(1 + \lfloor \log_2(1/s) \rfloor) \times k + \lceil d/2^{\lfloor \log_2(1/s) \rfloor} \rceil, \tag{19}$$

where k is the number of non-zero elements.

A.10. Additional Comparison with Other Baselines

In this section, we provide the additional comparison of our method against more baselines to supplement the empirical results of our work as shown below. The additional baselines include DASHA (Tyurin & Richtárik, 2023) and FedComGate (Haddadpour et al., 2021), both of which utilized the unbiased compressor, and PowerSGD (Vogels et al., 2019) which adopted a low-rank based approach. We evaluate methods on CIFAR100 dataset for ResNet18 under 0.6 Dirichlet distribution. Regarding the unbiased compressor for DASHA and FedComGate we use stochastic quantization. The quantization bit is set to be 4 in DASHA, and set to be 2 in FedComGate. The rank in PowerSGD is set to be 30. Note that we include momentum technique when implementing PowerSGD as done in their paper, which could provide an additional performance improvement. Unless stated otherwise, 'SB' in the table denotes the Sparse-Binary applied in the original space and 'Ours' refers to Mapping + Sparse-Binary. 'NA' stands for Not Achieved, meaning that the target accuracy never reached.

Table 5: The amount of bits (GiB) required to reach the target accuracy on CIFAR100 dataset

	SB (s=0.03)	SB (s=0.1)	DASHA	FedComGate	PowerSGD	Ours (s=0.03)	Ours (s=0.1)
Target=36%	0.06658	0.14137	1.20775	0.32939	1.54883	0.04481	0.10950
Target=40%	NA	0.36854	NA	0.67707	2.26146	0.07921	0.16588
Target=44%	NA	NA	NA	NA	NA	NA	0.40695

As can be seen in Table 5, our approach requires the fewest bits to reach each target accuracy, and it also achieves the highest accuracy among the baselines. As DASHA and PowerSGD did not consider severe heterogeneity or setups with a large scale of clients in their paper, these baselines fall short of our method in our setup. Moreover, while conventional unbiased compression-based methods (especially the methods that use stochastic quantization) have limitations in compression rate ($\times$ 32 in maximum), our approach combines the quantization with sparsification to demonstrate higher communication efficiency with better performance.

A.11. Discussion on Time Complexity

One may argue that the mapping approach requires a bit amount of time for constructing mapping, and transferring the gradient. In this section, we address concern regarding the time complexity and practicality of our approach in the following three aspects.

Firstly, we highlight that without requiring highly frequent periods for mapping construction, the performance of our approach remains solid. For the experiments in the main manuscript, we constructed the mapping every 20 rounds during the initial 100 rounds when evaluating our method. However, constructing mappings with such frequent period can reduce practicality in some cases. Thus, we have included an ablation study on the period of mapping construction as shown in Appendix A.8 (Figure 8). As can be seen, constructing the mapping once every 50 rounds, or even once every 100 rounds in our method, still allows us to reach a accuracy higher than, or comparable to the final accuracy of FedAvg in a communication-efficient manner.

Secondly, we measured the elapsed time taken for mapping construction and local training on the *same* GPU device in Table 3. However, this evaluation does not consider the actual difference in computing power between the server and clients. In fact, servers usually possess greater computing power compared to the participating nodes, which are typically resource constrained devices, such as smartphones, drones and IoT devices. Moreover, as set in the main results, only occasional construction also proves to be sufficient as the rounds progress (e.g. constructing mapping every 100 round from 300 to 600 rounds is sufficient).

Lastly, we can also consider an alternative option, called randomized SVD (RSVD) instead of SVD when computation resource is highly limited. The time complexity of RSVD is generally $O(mnk)$ with $n \times m$ matrix when obtaining k leading components, which could be much faster than that of vanilla SVD with time complexity of $O(\min(mn^2, nm^2))$. In fact, since we only need to accurately compute a few principal components which the vectors are mostly concentrated on, considering such efficient randomized algorithms would also work well in practice. Note that adopting randomized SVD does not change our algorithm at all; we simply replace the SVD process in our framework to randomized SVD as a tool for obtaining leading components. Since most of the SVD latency originates from the computation of the SVD on the input, RSVD is only performed for input activation (computation of leading components of output gradient was still performed using SVD). The following Table 6 shows the time (sec) taken for mapping construction, final accuracy, and required bits to reach target accuracy when using RSVD in comparison to SVD on CIFAR100 dataset (sparsity$= 0.03$). As can be seen, using RSVD allows for a reduction in time by more than 10%.

Table 6: Comparison of accuracy (%), elapsed time (sec), required bits (GiB) when constructing the mapping with SVD and RSVD

	Construction with SVD	Construction with RSVD
Time taken for mapping construction	47.9614	4.5782
Final Accuracy	42.597	41.057
Required bits (acc=40%)	0.07921	0.09688

We also provide the total local training time (sec) required to reach the target accuracy as shown in Table 7. We compare our method with FedAvg, Sparse-Binary, PowerSGD (Vogels et al., 2019) and ours with mapping constructed with RSVD. Here we only consider the local training time per client in measuring the time taken to reach the target accuracy. Note that for each compression method except for FedAvg, the total training time and the time per batch (or per round) includes the elapsed time for client-side compression. We also include the time taken to construct the mapping in our approach. We set rank as 30 for PowerSGD (including momentum technique), and set sparsity $= 0.1$ for for Sparse-Binary and ours. Unless stated otherwise, 'Ours' in the table refers to Mapping + Sparse-Binary. The target accuracy is 40%.

Table 7: Comparison of total time taken to reach the target accuracy, time per batch and time per round across baselines

	FedAvg	Sparse-Binary	PowerSGD	Ours (SVD)	Ours (RSVD)
Time per batch	0.0565	0.0642	0.0573	0.0992	0.0819
Time per round	2.8248	3.2111	2.8628	4.9622	4.0945
Time to reach target acc	983.05	1448.20	1571.66	1197.44	1051.11

As rounds progress, since the mapping period becomes less frequent, taking the average per round overall does not result in a significant difference, and this is even more the case when using RSVD to construct the mapping. Moreover, when considering the time it takes to reach the target accuracy, our method is faster than some baselines, which demonstrates that our approach is capable of achieving both desired performance and speed. This suggests that, although it takes longer time per round compared to other methods, it is not problematic in terms of satisfying the target accuracy. Additionally, we note that we have not yet considered the asymmetry in computation power between the client and the server. If we take this into account, the time per round would also become increasingly negligible.

A.12. Supplemental Experiments for Main Results

In this section, we supplement main results with additional experiments. These results are evaluated based on varying Dirichlet parameters α, compression error $1-\rho$ for Top-k, sparsity level s for kSB, and across various datasets. The following figures (Figure 9 - 16) illustrate the accuracy over communication rounds and bits. As can be seen, better enhancements in communication efficiency compared to baselines can be observed in most cases, which is consistent with the main plot.

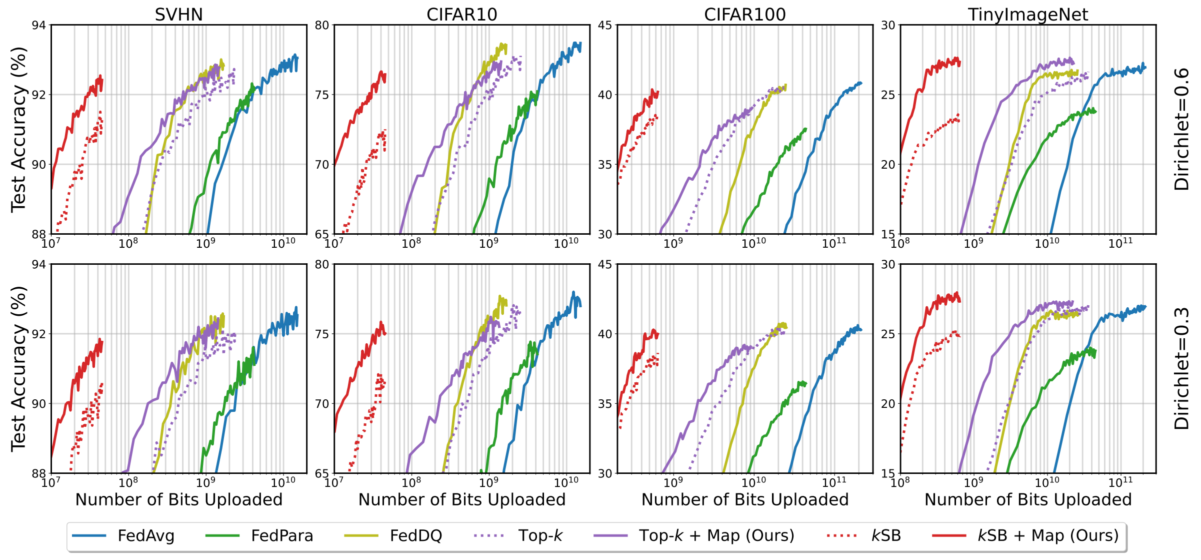


Figure 9: Accuracy versus number of communication bits across various benchmarks, including SVHN, CIFAR10, CIFAR100 and TinyImageNet under Dirichlet distribution with $\alpha = 0.6$ **(top row)** and $\alpha = 0.3$ **(bottom row)** among clients ($s = 0.01$, $\rho = 0.6$).

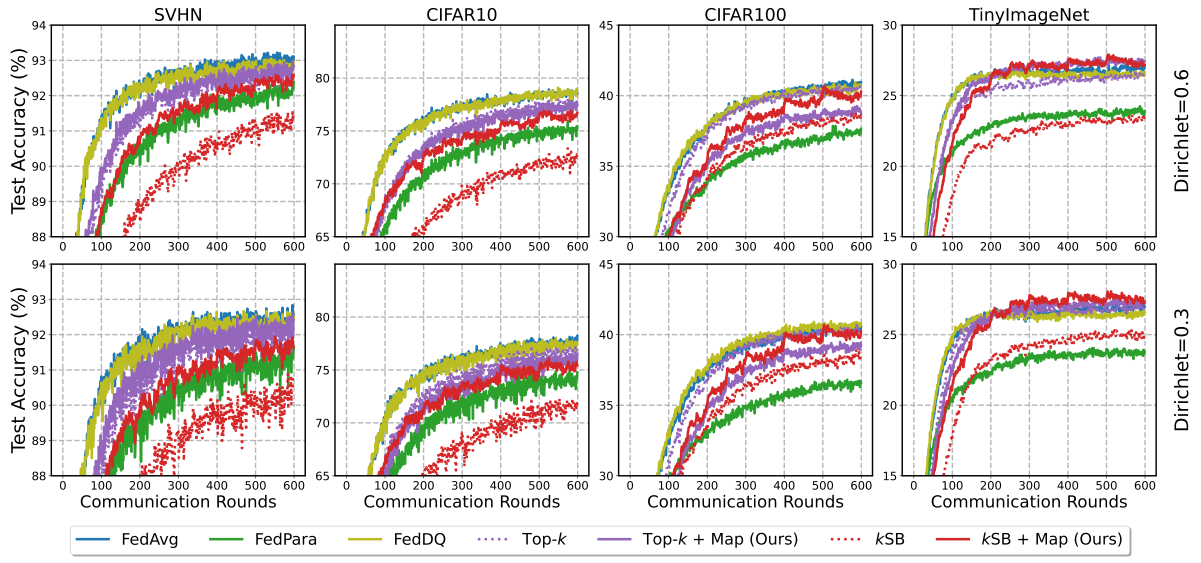


Figure 10: Accuracy versus communication rounds across various benchmarks, including SVHN, CIFAR10, CIFAR100 and TinyImageNet under Dirichlet distribution with $\alpha = 0.6$ **(top row)** and $\alpha = 0.3$ **(bottom row)** among clients ($s = 0.01$, $\rho = 0.6$).

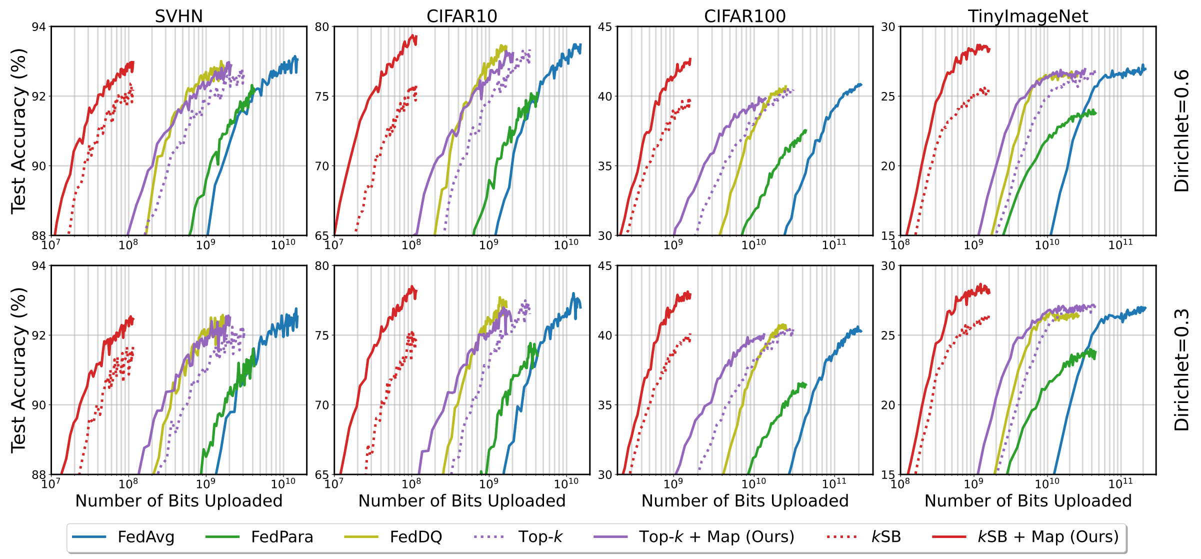


Figure 11: Accuracy versus number of communication bits across various benchmarks, including SVHN, CIFAR10, CIFAR100 and TinyImageNet under Dirichlet distribution with $\alpha = 0.6$ **(top row)** and $\alpha = 0.3$ **(bottom row)** among clients ($s = 0.03$, $\rho = 0.7$).

A.13. Supplemental Results for PD Baselines

In this section, we supplement the additional PD baseline results (Figure 17 - 24). As previously mentioned, we evaluated these results based on various α, ρ, s across various datasets. These findings are consistent with the selective results in the main manuscript, indicating that all the PD baselines fail to enhance the existing compressors. Given this, our approach is still the only method among those utilizing public data that improves performance.

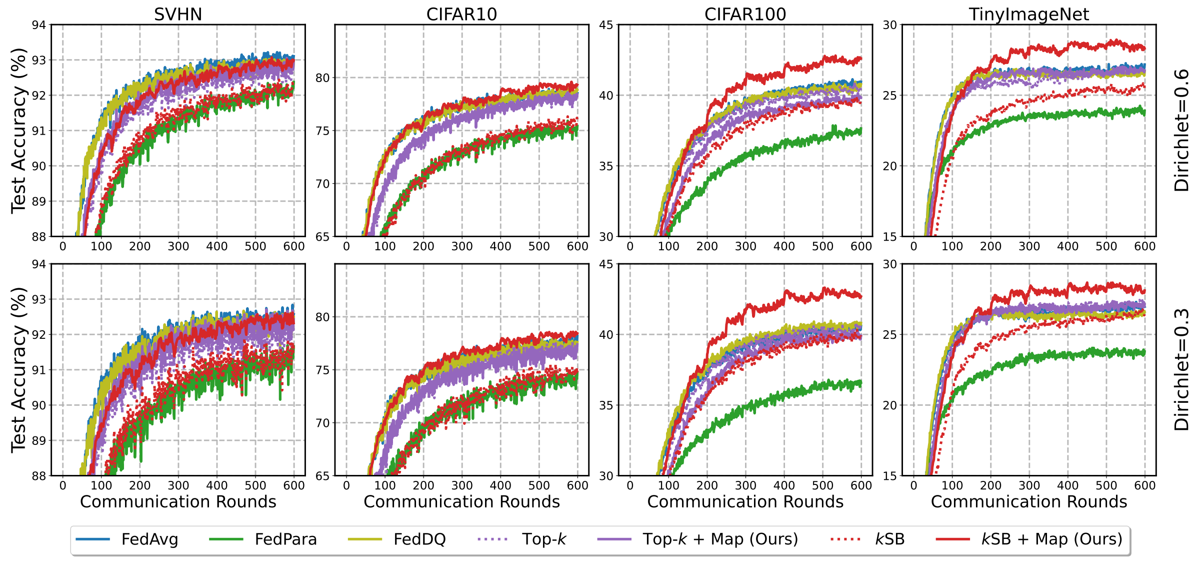


Figure 12: Accuracy versus communication rounds across various benchmarks, including SVHN, CIFAR10, CIFAR100 and TinyImageNet under Dirichlet distribution with $\alpha = 0.6$ **(top row)** and $\alpha = 0.3$ **(bottom row)** among clients ($s = 0.03$, $\rho = 0.7$).

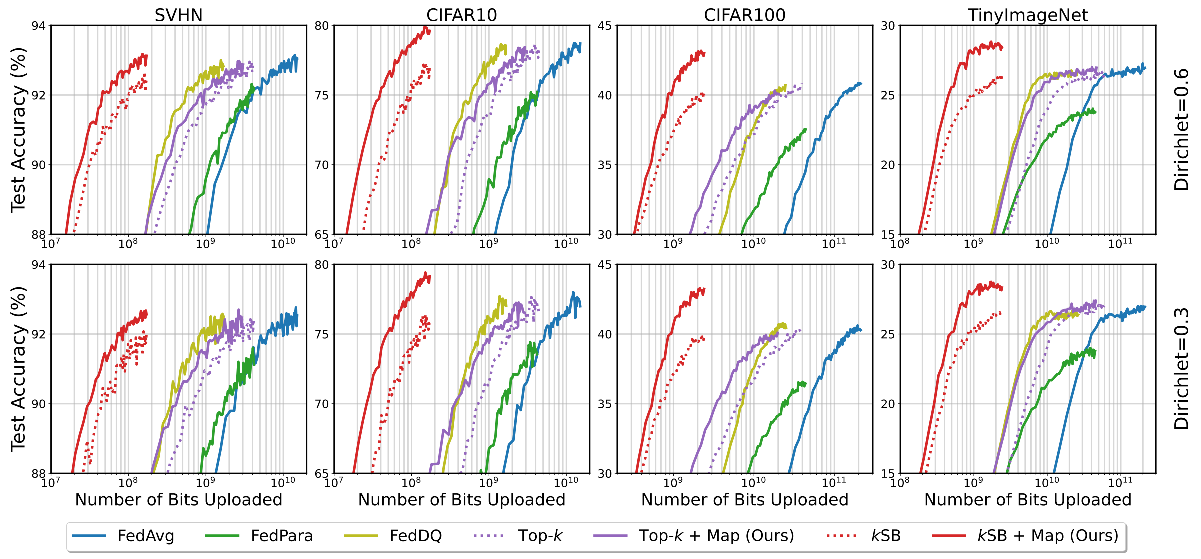


Figure 13: Accuracy versus number of communication bits across various benchmarks, including SVHN, CIFAR10, CIFAR100 and TinyImageNet under Dirichlet distribution with $\alpha = 0.6$ **(top row)** and $\alpha = 0.3$ **(bottom row)** among clients ($s = 0.05$, $\rho = 0.8$).

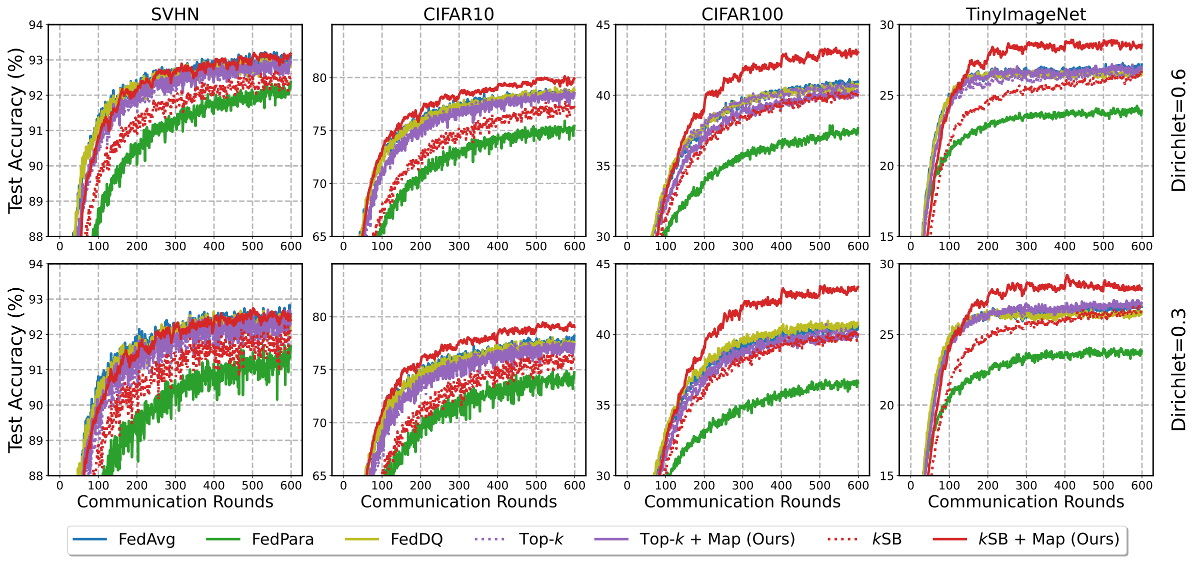


Figure 14: Accuracy versus communication rounds across various benchmarks, including SVHN, CIFAR10, CIFAR100 and TinyImageNet under Dirichlet distribution with $\alpha = 0.6$ **(top row)** and $\alpha = 0.3$ **(bottom row)** among clients ($s = 0.05$, $\rho = 0.8$).

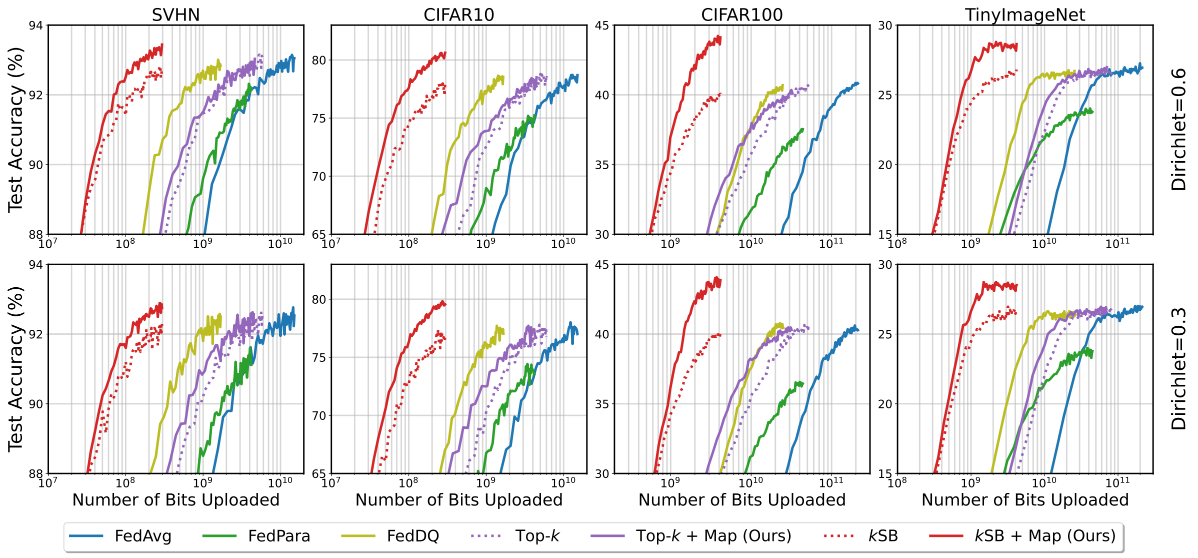


Figure 15: Accuracy versus number of communication bits across various benchmarks, including SVHN, CIFAR10, CIFAR100 and TinyImageNet under Dirichlet distribution with $\alpha = 0.6$ **(top row)** and $\alpha = 0.3$ **(bottom row)** among clients ($s = 0.1$, $\rho = 0.9$).

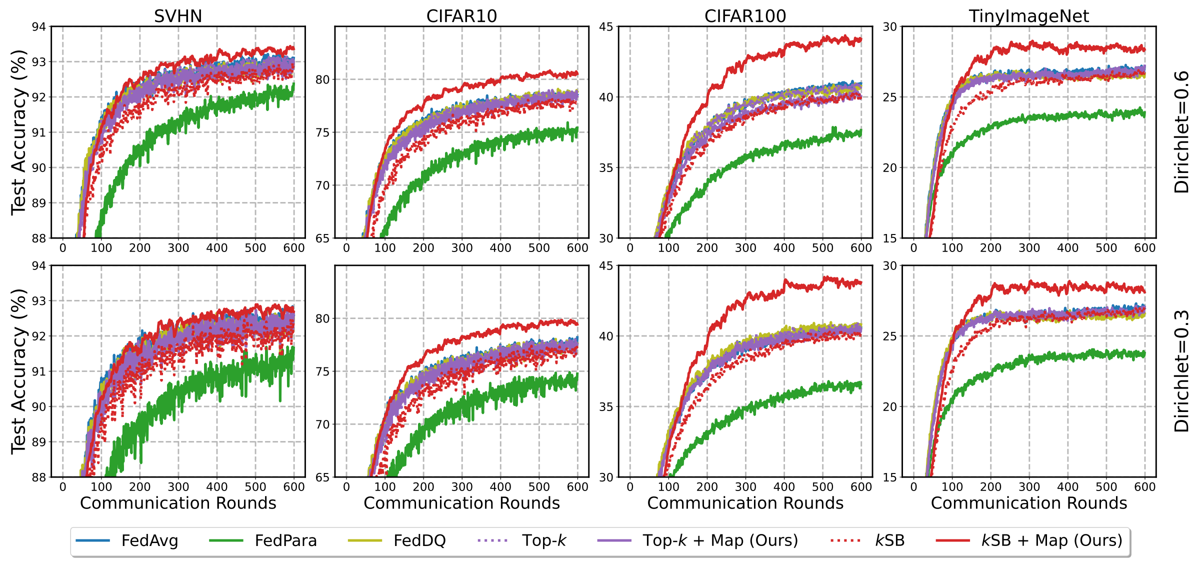


Figure 16: Accuracy versus communication rounds across various benchmarks, including SVHN, CIFAR10, CIFAR100 and TinyImageNet under Dirichlet distribution with $\alpha = 0.6$ **(top row)** and $\alpha = 0.3$ **(bottom row)** among clients ($s = 0.1$, $\rho = 0.9$).

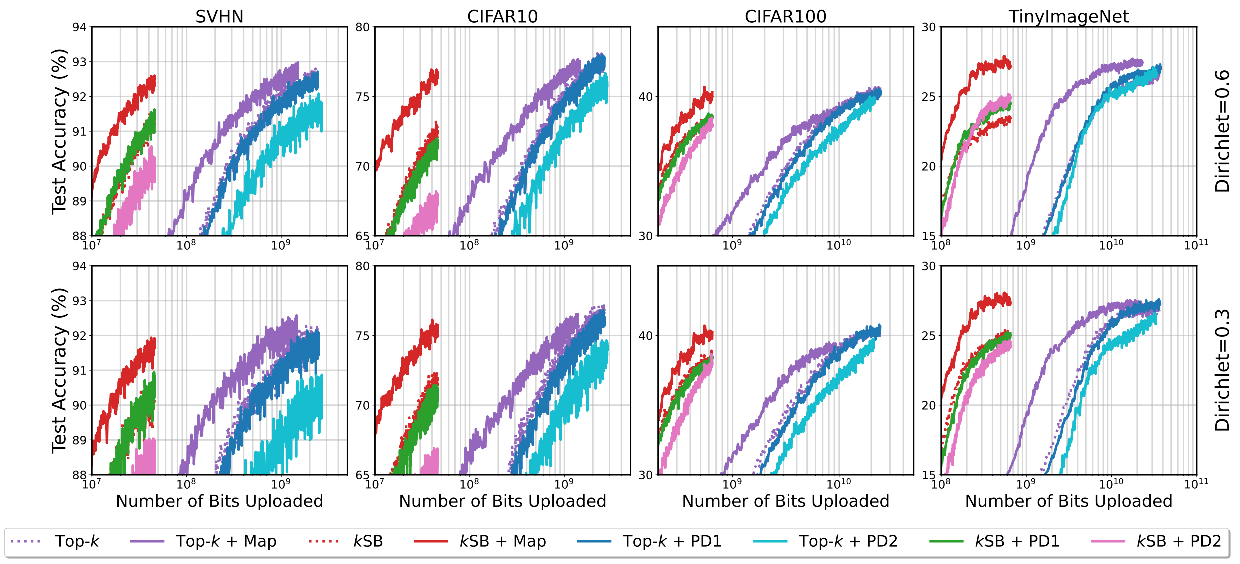


Figure 17: Comparison results with PD baselines for accuracy versus communication bits across various benchmarks, including SVHN, CIFAR10, CIFAR100 and TinyImageNet under Dirichlet distribution with $\alpha = 0.6$ **(top row)** and $\alpha = 0.3$ **(bottom row)** among clients. ($s = 0.01$, $\rho = 0.6$).

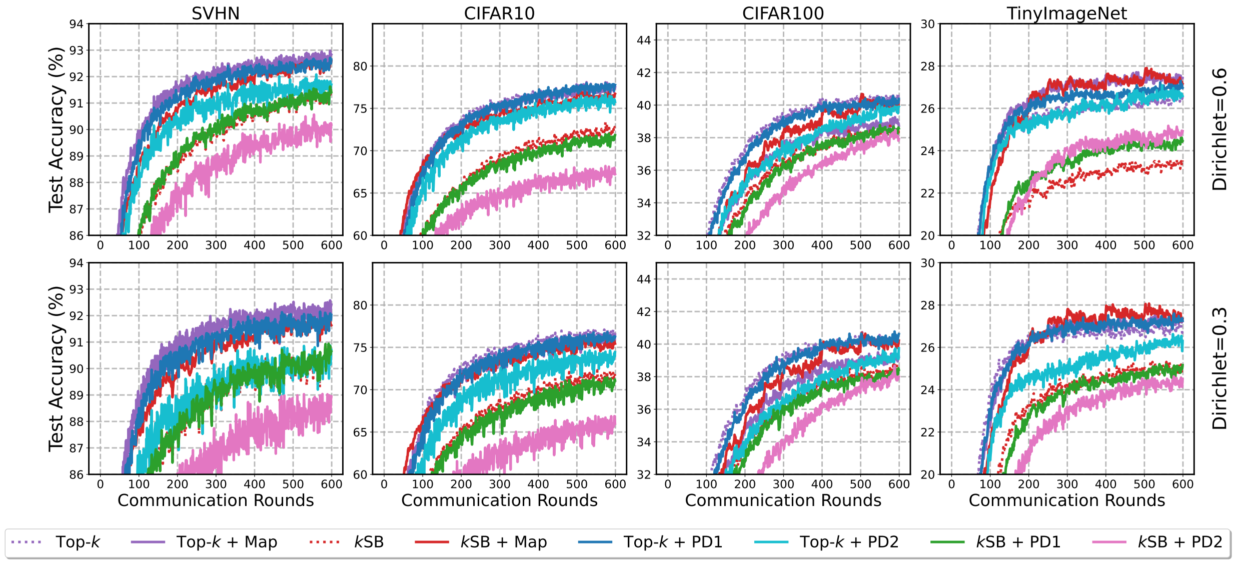


Figure 18: Comparison results with PD baselines for accuracy versus communication rounds across various benchmarks, including SVHN, CIFAR10, CIFAR100 and TinyImageNet under Dirichlet distribution with $\alpha = 0.6$ **(top row)** and $\alpha = 0.3$ **(bottom row)** among clients. $(s = 0.01, \rho = 0.6)$.

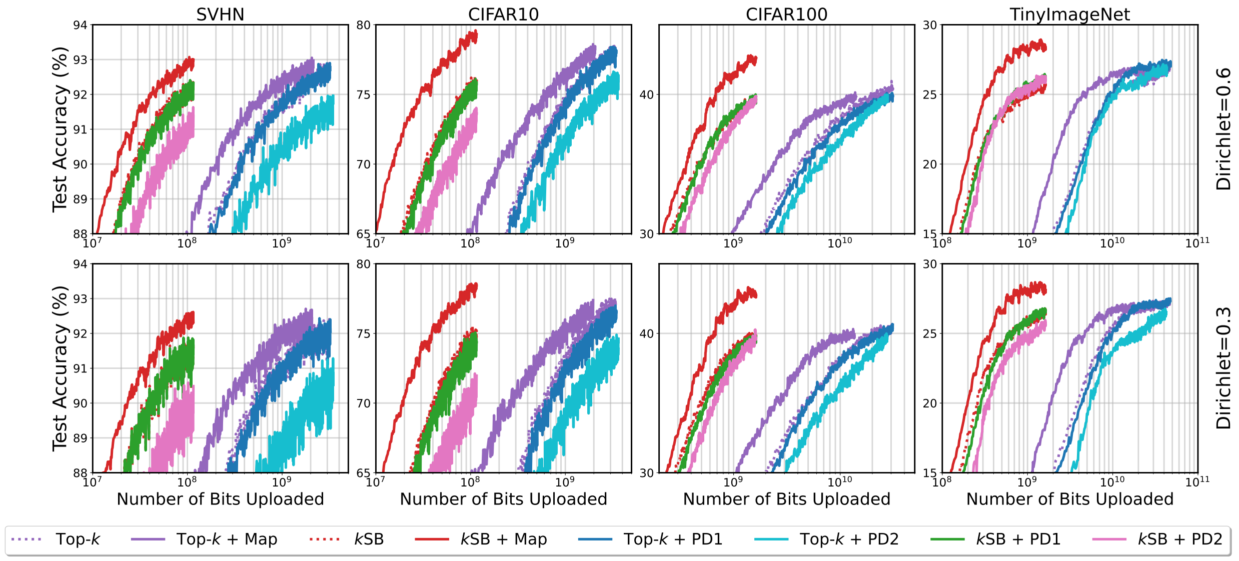


Figure 19: Comparison results with PD baselines for accuracy versus communication bits across various benchmarks, including SVHN, CIFAR10, CIFAR100 and TinyImageNet under Dirichlet distribution with $\alpha = 0.6$ **(top row)** and $\alpha = 0.3$ **(bottom row)** among clients. $(s = 0.03, \rho = 0.7)$.

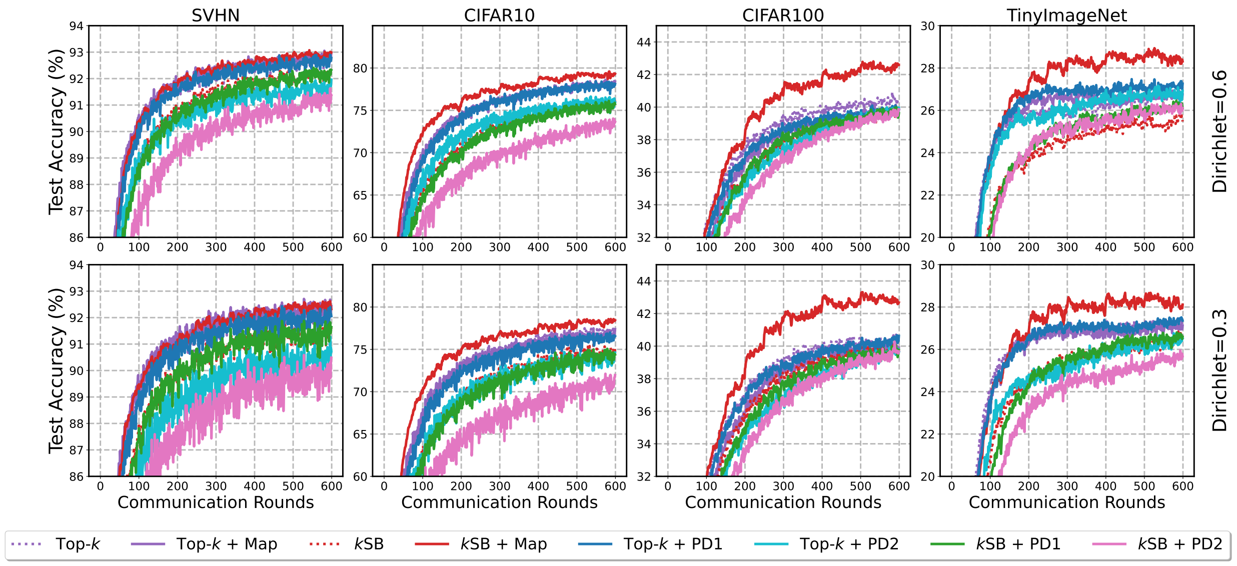


Figure 20: Comparison results with PD baselines for accuracy versus communication rounds across various benchmarks, including SVHN, CIFAR10, CIFAR100 and TinyImageNet under Dirichlet distribution with $\alpha = 0.6$ **(top row)** and $\alpha = 0.3$ **(bottom row)** among clients. ($s = 0.03$, $\rho = 0.7$).

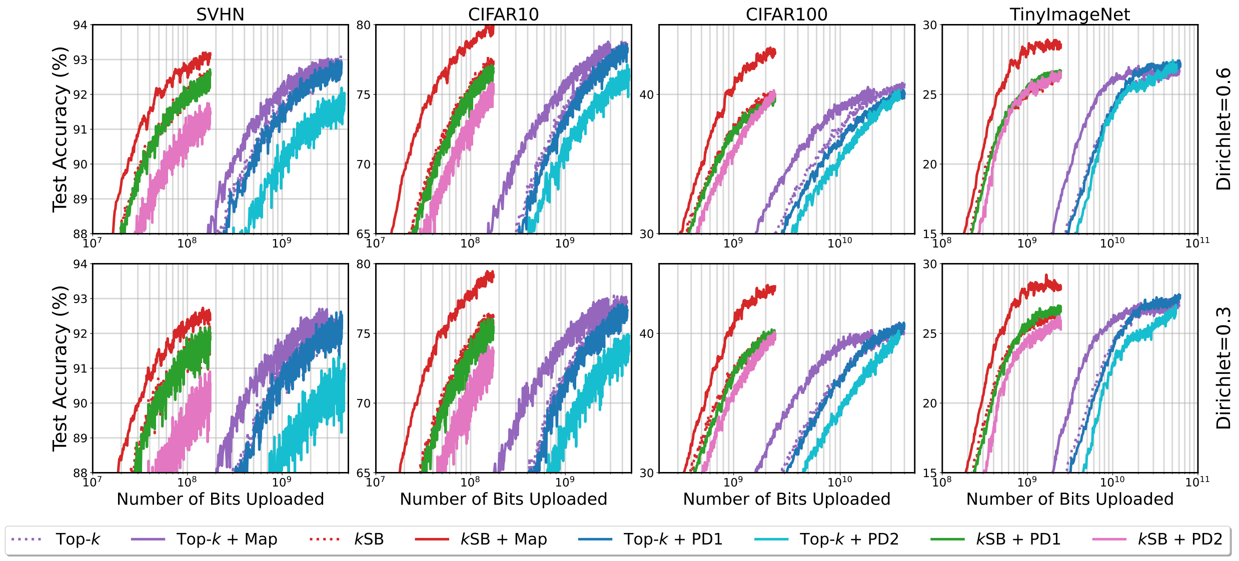


Figure 21: Comparison results with PD baselines for accuracy versus communication bits across various benchmarks, including SVHN, CIFAR10, CIFAR100 and TinyImageNet under Dirichlet distribution with $\alpha = 0.6$ **(top row)** and $\alpha = 0.3$ **(bottom row)** among clients. ($s = 0.05$, $\rho = 0.8$).

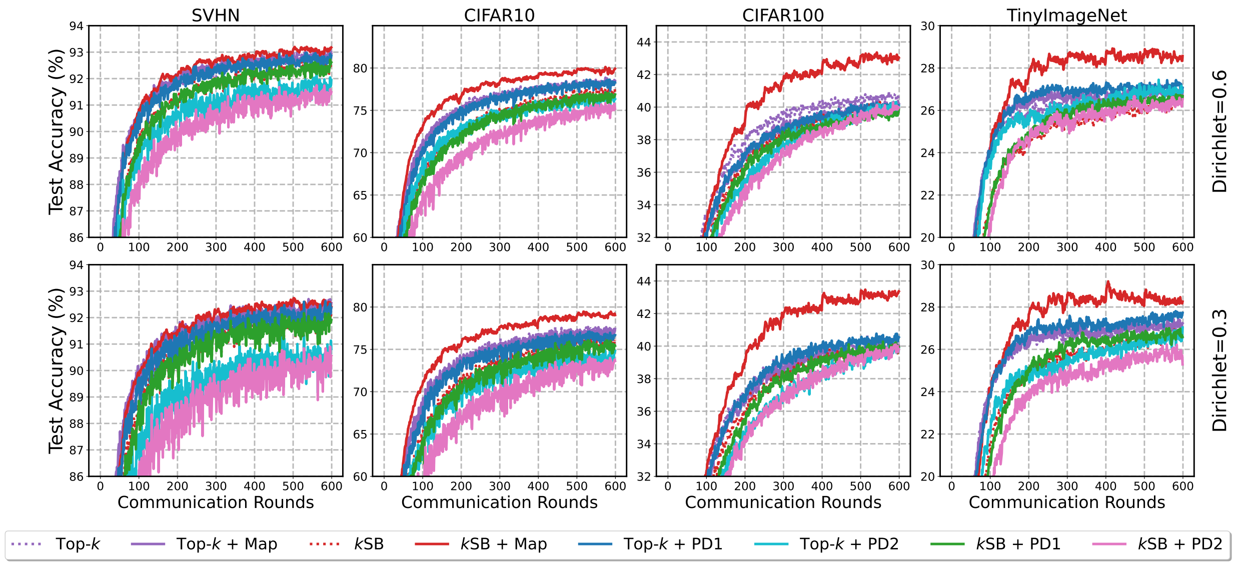


Figure 22: Comparison results with PD baselines for accuracy versus communication rounds across various benchmarks, including SVHN, CIFAR10, CIFAR100 and TinyImageNet under Dirichlet distribution with $\alpha = 0.6$ **(top row)** and $\alpha = 0.3$ **(bottom row)** among clients. $(s = 0.05, \rho = 0.8)$.

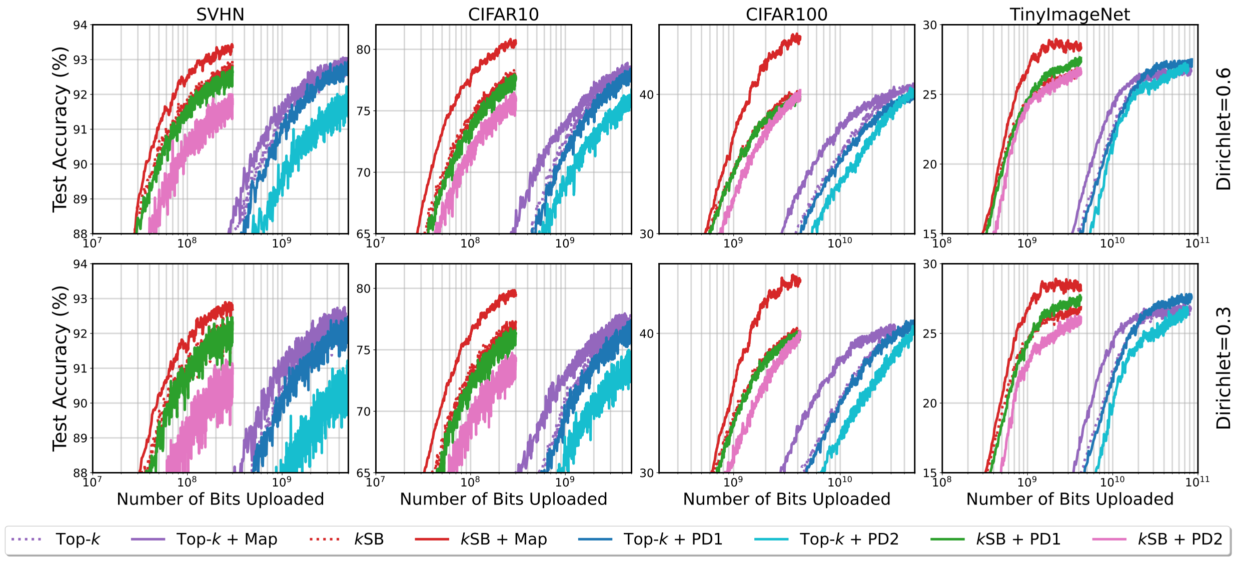


Figure 23: Comparison results with PD baselines for accuracy versus communication bits across various benchmarks, including SVHN, CIFAR10, CIFAR100 and TinyImageNet under Dirichlet distribution with $\alpha = 0.6$ **(top row)** and $\alpha = 0.3$ **(bottom row)** among clients. $(s = 0.1, \rho = 0.9)$.

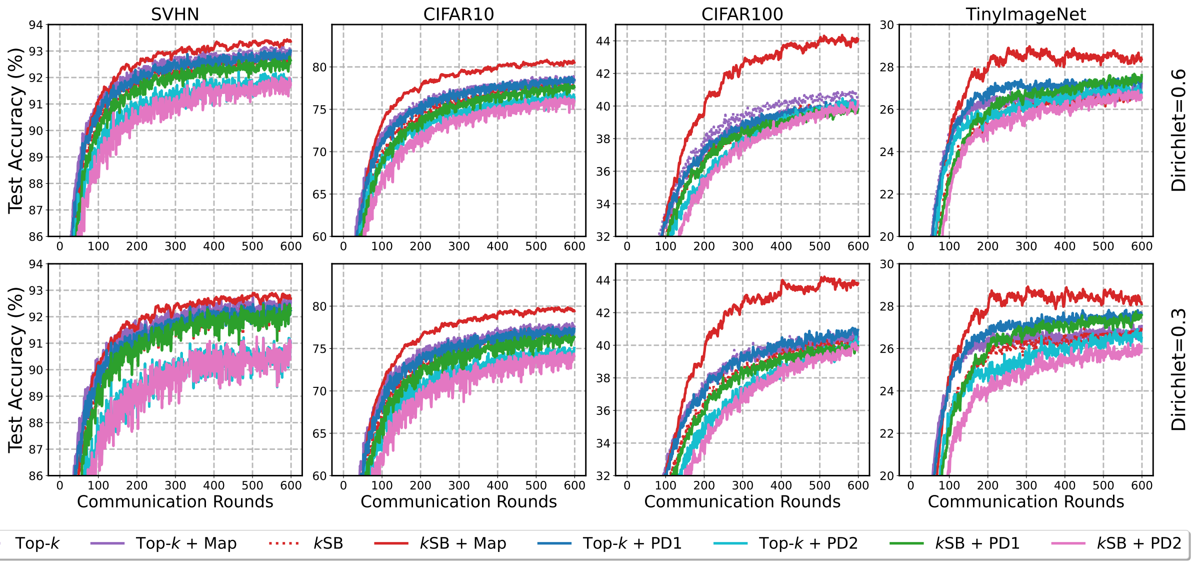


Figure 24: Comparison results with PD baselines for accuracy versus communication rounds across various benchmarks, including SVHN, CIFAR10, CIFAR100 and TinyImageNet under Dirichlet distribution with $\alpha = 0.6$ **(top row)** and $\alpha = 0.3$ **(bottom row)** among clients. ($s = 0.1$, $\rho = 0.9$).