Fed2Com: Towards Efficient Compression in Federated Learning

Yu Zhang[1]*, Wei Lin[1]*, Sisi Chen[1], Qingyu Song[1], Jiaxun Lu[2], Yunfeng Shao[2], Bei Yu[1], and Hong Xu[1]
[1]Dept. of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong.
[2]Huawei Technologies Noah's Ark Lab, China.

***Abstract*—Federated learning (FL) is a distributed machine learning system that enables multiple clients to collaboratively train a machine learning model without sacrificing data privacy. In the last few years, various biased compression techniques have been proposed to alleviate the communication bottleneck in FL. However, these approaches rely on an ideal setting where all clients participate and continuously send their local errors to the cloud server. In this paper, we design a communication-efficient algorithmic framework called Fed2Com for FL with non-i.i.d datasets. In particular, Fed2Com has a two-level structure: At the client side, it leverages unbiased compression methods, e.g., rand-k sparsification, to compress the upload communication, avoiding leaving errors at the client. Then on the server side, Fed2Com applies biased compressors, e.g., top-k sparsification, with error correction to compress the download communication while stabilizing the training process. Fed2Com can achieve high compression ratio while maintaining robust performance against data heterogeneity. We conduct extensive experiments on MNIST, CIFAR10, Sentiment140 and PersonaChat datasets, and the evaluation results reveal the effectiveness of Fed2Com.**

***Index Terms*—Federated learning, Communication compression, Non-i.i.d data**

I. Introduction

Traditionally, the training of machine learning models relies on a large data center to collect and process a vast amount of client data, which incurs high latency, low bandwidth, and privacy/security concerns. In this circumstance, federated learning (FL) has emerged as a prevailing setting for training machine learning models as it allows multiple clients to collaboratively train a model without data sharing [1]. In the federated learning settings, participating clients do not send their local data to the cloud server. Instead, they perform multiple epochs of SGD based on their local datasets and communicate their models to the cloud. Then, the cloud server coordinates an optimization procedure among the clients and updates the global model parameters. By doing so, not only can FL significantly alleviate the risk of compromising privacy/security, but it also reduces communication costs through infrequent communication.

Despite the aforementioned advantages of FL, FL as it stands now faces two dire challenges: high communication overhead due to the ever-increasing sizes of learning models [2], and non-i.i.d data across clients that makes convergence slow or even not possible. The two seem to be at odds with each other, as intuitively sharing less data for training means less potential or information to achieve convergence. Some recent work suggests that sparsification schemes like top-k can address these two challenges simultaneously [3]. The caveat is that error feedback is needed to ensure convergence [3]. In particular, error feedback works by storing the compression error in local memory and adding it to the global updates in the next iteration. This assumes an ideal setting where all clients participate and continuously send their local errors in each round, which unfortunately does not hold in practical FL settings. With a large number of clients, usually just a small fraction of them are chosen randomly in each round to send their local updates to the central server(s). The seminal FedAvg [1] along with many other FL systems follow this setup [2], [4], [5]. In this case, error feedback does not work because very likely a client will not be chosen enough times for its errors to be meaningfully accumulated [2], which implies that top-k may even lead to divergence [6] (more details in §III-A).

In this work, we propose a novel framework that can address both the communication and convergence challenges in practical FL settings with sparse clients. Our framework, dubbed Fed2Com, has a two-level structure: At the client side, a participating client uses unbiased compression methods, e.g., rand-k sparsification, to compress its local updates. Particularly, unbiased compressors can reduce the amount of data to be sent to the cloud, and combat the slow convergence effect that biased methods such as top-k bring without error feedback. Then at the server side in the cloud, to further reduce the communication overhead, we apply biased compressors, e.g., top-k sparsification, to dispatch only the most significantly updated parameters to the clients. Since the biased compression is now applied to the model parameters, the biased errors naturally accumulate, and the error correction method can work now. We do not use unbiased compressors here as it has been shown that biased compression techniques with error correction are more effective in stabilizing overall convergence [3]. Naturally, with unbiased compressors at client side and biased compressors at cloud side, Fed2Com can achieve both uplink and downlink compression, thus solving the communication challenge.

We perform several numerical experiments which validate the effectiveness of our proposed Fed2Com. Under the regime of infrequent communication and non-i.i.d datasets, our proposed Fed2Com could achieve a high compression ratio while maintaining fast convergence, whereas FedAvg suffers from

* These authors contribute equally to this work.

979-8-3503-7099-7/24/$31.00 ©2024 IEEE

severe performance deterioration or even failure of training. In particular, we find the dual-compression module, i.e., unbiased compressors at client and biased compressors at cloud, contributes to not only the bidirectional communication compression but also the stabilization of the training process.

In general, our contribution can be summarized as follows:

- We find out that utilizing unbiased compression at the client side and biased compression with error feedback at the cloud side significantly enhances FL framework convergence in the context of non-i.i.d settings, thus offering a robust solution to data heterogeneity.
- We propose Fed2Com, a simple yet effective FL framework, to achieve efficient communication and fast convergence in the regime of small local data and heterogeneous datasets.
- We empirically verify the effectiveness of our method with two image recognition tasks and two language modeling tasks. Particularly, we find that Fed2Com can reduce up to $\frac{1}{13000}$ of the uplink communication with minimal impact on learning accuracy.

II. Related Work

For communication-constrained environments, there are many works on decreasing the communication overhead. In general, these works can be categorized into two classes: one is to reduce the communication rounds and the other is to compress the uplink/downlink transmission in each communication round. Here, we briefly introduce the ones that are close to our work.

Infrequent Communication: One notable algorithm to reduce the total number of bytes transferred during training is FedAvg [1], where clients carry out multiple steps of stochastic gradients descent (SGD) locally before sending the model update to the aggregator. However, one disadvantage of FedAvg is that taking many local steps can result in degraded convergence on non-i.i.d data. This makes intuitive sense since taking many local steps on client datasets that are not representative of the overall distribution would lead to local overfitting, hence hindering global convergence [4]. In other words, although FedAvg has a convergence guarantee for the i.i.d setting [7], this guarantee can not be applied directly to the non-i.i.d setting. Variants of FedAvg have been proposed to improve its performance on non-i.i.d data. Sahu et al. [5] propose FedProx to constrain the local gradient update steps in FedAvg by penalizing the L2 distance between local models and the current global model. Under the bounded dissimilarity assumption, FedProx could recover the convergence rate of SGD. Luping et al. [8] propose Communication-Mitigated Federated Learning (CMFL) to dynamically identify the irrelevant updates made by clients and preclude them from being uploaded in advance [8]. In this way, CMFL can substantially reduce the communication overhead by decreasing the communication rounds while still guaranteeing the learning convergence on non-i.i.d setting [8].

Parameter Compression: Even with a satisfiable convergence rate on non-i.i.d datasets, these methods require local clients to upload or download an intact model. More specifically, in each communication round, clients have to download and upload the whole model parameters, which would pose a heavy burden on local clients since clients typically connect to the central aggregator over slow and unreliable connections ($\sim$1Mbps) [9]. One natural solution to solve this problem is to compress the stochastic gradient such that the result is still an unbiased estimate of the true gradient, including stochastic quantization [10] and stochastic gradient sparsification [11]. However, unbiased compressors might introduce extra variance [12]. Combined with error feedback, biased compressor, such as top-k [13] sparsification and signSGD [14], can often achieve superior performance when compared to their unbiased counterparts, which is attributed to their lower empirical variance [12]. Yang et al. [3] propose to leverage biased compressor, i.e., top-k, with error feedback at client side, which only slightly increases the local variance constant and does not affect the overall convergence rate for non-i.i.d. FL with infrequent communication. Nevertheless, carrying out error feedback requires local client states, which is often infeasible in federated learning since the local error may not be able to be re-introduced due to random client selection.

III. Motivation

In this section, we motivate our work based on two examples: Through the first, we illustrate the necessity to perform unbiased compression at client side; and through the second we demonstrate the superiority of carrying out biased compression at cloud side.

A. Unbiased Compression at Client

Here we train a ResNet9 [15] model (with 6.5M parameters) with CIFAR10 [16] in the FL framework with both i.i.d and non-i.i.d datasets. In particular, although CIFAR10 is an i.i.d dataset, we create a non-i.i.d dataset by assigning each client images from only a single class without overlapping. We set 10,000 clients, each having 5 images, and one cloud server. We train with 1% clients participating in each iteration, for a total of 96 iterations. Besides, each selected client performs 4 local steps before sending updates to the cloud. We examine both biased and unbiased compression methods on the client side. Particularly, the biased compression method performed at the client side is top-k with error feedback, and the unbiased compression technique is rand-k. Note that in top-k, we do not suppose clients will be selected enough times to upload compression errors to the cloud. Since clients are chosen randomly, chances are that they will not be able to upload their local errors until the end of training. For rand-k, to ensure unbiasedness, we randomly drop out coordinates of the gradients and proportionally scale the remaining coordinates by $\frac{d}{k}$ [11].

Fig. 1 shows the training loss and accuracy of different FL schemes. In the i.i.d setting in Fig. 1(a) and 1(b), all methods

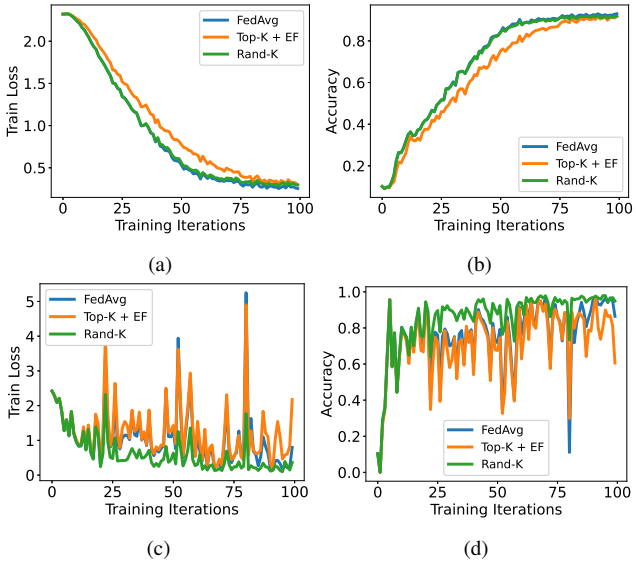


Fig. 1. Comparison of top-k + EF and rand-k sparisification at client side. (a) ResNet9 training loss curve under i.i.d setting. (b) ResNet9 training accuracy rurve under i.i.d setting (c) ResNet9 training loss curve under non-i.i.d setting. (d) ResNet9 training accuracy rurve under non-i.i.d setting.

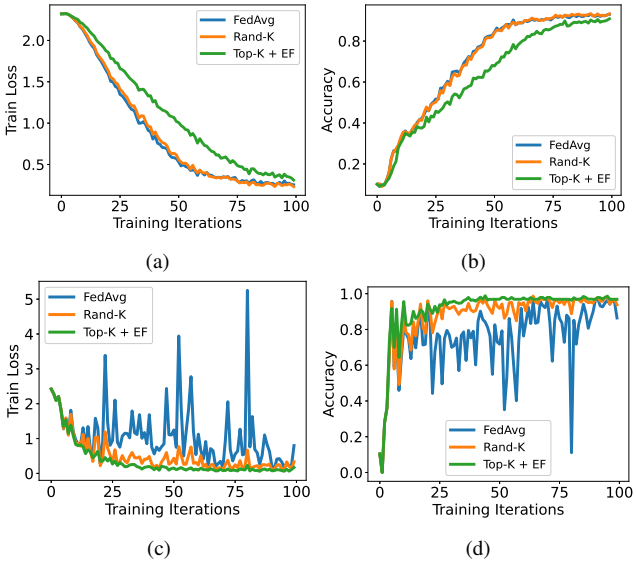


Fig. 2. Comparison of top-k (with error feedback) and rand-k sparisification at cloud side. (a) ResNet9 training loss curve under i.i.d setting. (b) ResNet9 training accuracy curve under i.i.d setting. (c) ResNet9 training loss curve under non-i.i.d setting. (d) ResNet9 training accuracy curve under non-i.i.d setting

show signs of convergence and reach good accuracy. Particularly, rand-k enjoys comparable performance as uncompressed FedAvg, and outperforms top-k with faster convergence and higher accuracy. In the non-i.i.d setting in Fig. 1(c) and 1(d), both FedAvg and top-k suffer significant performance degradation with poor convergence and diverging accuracy. Rand-k, however, exhibits robustness and better performance, potentially due to the stochasticity introduced by its compression, which could act as a form of regularization. This could prevent overfitting to each client's local data and promote the learning of more universal features, thereby improving learning robustness amidst data heterogeneity.

In general, the unbiased rand-k outperforms the biased top-k with EF on both i.i.d and non-i.i.d datasets, demonstrating the necessity of unbiased compressing at the client-side.

B. Biased Compression at Cloud

To further reduce the communication overhead, we also wish to compress the downlink data from cloud side to client side. Instead of using unbiased compressors, we choose biased compression methods. This does not conflict with our previous arguments about the limitations of error feedback on biased compression. The rationale is that biased compression is implemented at the cloud across the entire set of updated model weights, and therefore, the error term naturally envelops all weights. As such, it doesn't encounter the sparse client problem when deployed to clients. Further, with error feedback, biased compressors can stabilize framework convergence with a lower variance of gradients in the regime of small and non-i.i.d local datasets, and outperform unbiased compressors [6], [12]. Observe from Fig. 2 that though top-k with error feedback performs the worst in the i.i.d setting, it demonstrates its superiority in stabilizing the training process with better accuracy in the non-i.i.d setting as depicted in Fig. 2(c) and 2(d).

In the non-i.i.d setting, each client is working with their own subset of data, which may be quite different from each other. Some of this variance could be considered as 'noise' with respect to the overall learning task. By applying top-k compression, the system might be focusing on the strongest gradients, i.e., the most 'important' learning signals. This could inherently filter out some of the client-specific noise, hence making the system more robust against data heterogeneity. In addition, the use of top-k compression could encourage convergence by selectively updating the most relevant model parameters. As each client will be updating and communicating only the most significant parameters, these global updates will tend to be more consistent across clients, even in the presence of heterogeneous data. Consequently, this can lead to more stable and faster convergence.

Based on our motivating examples, we demonstrate the necessity to perform unbiased compression at client side as well as the effectiveness of biased compressors.

IV. Design

We present the design of Fed2Com in this section.

A. Problem Setup

We first present the problem formulation along with the notations used throughout this paper. Consider a federated learning system with m clients who share the same loss function and coordinately train a model with distributed data. Each client i hosts a dataset with data distribution D_i. In practice, the client's local dataset is generated from its own

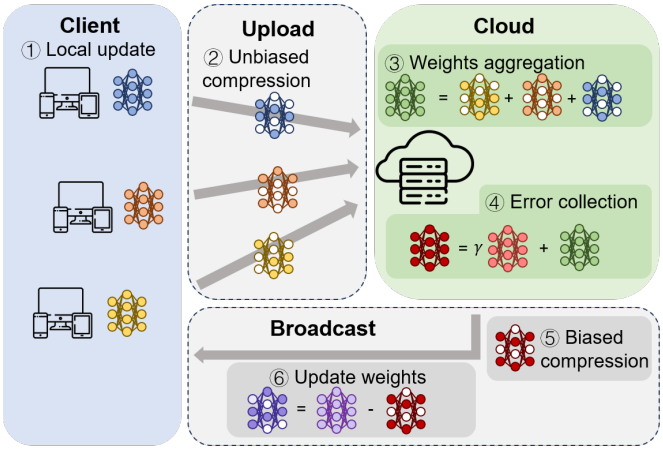


Fig. 3. Fed2Com overview.

environment and hence *non-i.i.d*, i.e., $D_i \neq D_j$, if $i \neq j$. To train an ML model, we in fact solve the following optimization problem:

$$\min_{x \in \mathbb{R}^d} f(x) := \frac{1}{m} \sum_{i=1}^{m} f_i(x),$$

where $f_i(x) \triangleq E_{\psi_i \sim D_i}[f_i(x, \psi_i)]$ denotes the local (possibly non-convex) loss function and the random training samples ψ_i follows the local data distribution D_i. Note that the i.i.d setting where each local dataset is sampled from the common latent distribution, i.e., $D_i = D, \forall i \in [m]$, can naturally be viewed as a special case of the non-i.i.d setting. Hence, our results for the non-i.i.d setting are directly applicable to i.i.d setting.

B. Design

Fed2Com is a two-level framework as depicted in Fig. 3. We aim to not only reduce the downlink and uplink communication overhead but also to guarantee good convergence on non-i.i.d datasets. The framework contains three key stages:

1. *Local Training:* In Fed2Com, each selected client i first performs several local steps and computes weight updates based on local data (step 1 in Fig. 3). Then, the updates are compressed with an unbiased compression method $\mathcal{UC}(\cdot)$ (step 2), and uploaded to the server.

2. *Global Update:* Upon receiving all updates from clients, the cloud server directly aggregates the local updates (step 3) to obtain global updates. It performs Polyak's momentum, which is known to achieve fast convergence and good generalization [17]. As biased compression introduces bias, the server applies the error correction technique, which entails that it adds to the global updates the difference between the compressed and original update vectors calculated in the previous round (step 4). Then we select the most significant global updates, i.e., perform biased compression to global updates, to update the parameters of the global model (step 5).

Algorithm 1: Fed2Com

Input: Initialize local weights x_i^0 for each client $i \in \{1, 2, ..., m\}$, error term $e^1 = 0$, and momentum buffer $u^0 = 0$.
Input: Local learning rate $\gamma_l > 0$, global learning rate $\gamma > 0$, momentum coefficient $\rho \in [0, 1)$, number of timesteps T, number of clients selected per round n, local batchsize l, local number of epochs K, unbiased compressor $\mathcal{UC}$, biased compressor $\mathcal{BC}$.
Output: $\{x^t\}_{t=1}^T$

for $t = 1, 2, ..., T$ **do**
  Randomly select n clients $c_1, ..., c_n$ out of m
  do in parallel at each client
    Download sparse weight update $x^t - x^0$
    for *local epoch* $p = 1, ..., K$ **do**
      Compute an unbiased stochastic gradient estimate $\xi_i^t = \frac{1}{l} \sum_{j=1}^{l} \nabla f_i(x^t, d_j)$
      Local update: $x_i^{t,p+1} = x_i^{t,p} - \gamma_l \xi_i^t$
    end
    $g_i^t = \mathcal{UC}(x_i^{t,K+1} - x_i^{t,1})$ `// Unbiased compression`
    Upload g_i^t to the cloud
  end
  $g^t = \frac{1}{n} \sum_{i=1}^{n} g_i^t$ `// Aggregate updates`
  $u^t = \rho u^{t-1} + g^t$ `// Momentum`
  $\delta^t = \gamma u^t + e^t$ `// Error correction using error up to the previous round`
  $\triangle^t = \mathcal{BC}(\delta^t)$ `// Biased compress`
  $e^{t+1} = \delta^t - \triangle^t$ `// Error accumulation`
  $x^{t+1} = x^t - \triangle^t$ `// Update weights`
end

Layer Type	Size
Convolution + ReLu	5 × 5 × 32
Max Pooling	2 × 2
Convolution + ReLu	5 × 5 × 64
Max Pooling	2 × 2
Fully Connected + ReLu	7 × 7 × 64
Fully Connected	10 × 64

TABLE I
CNN architecture for MNIST

3. *Model Distribution:* Finally, Fed2Com updates the k selected weights, and sends to the newly selected clients the delta between the current and their initial models (step 6), thus achieving the downlink compression.

Algorithm 1 summarizes the details of Fed2Com.

V. Evaluation

In this section, we perform extensive experiments to evaluate Fed2Com. In particular, our evaluation aims to answer the following questions:

Model	Dataset	Method	k in rand-k	k in top-k	Accuracy	Download Compression	Upload Compression	Total Compression
CNN	MNIST	FedAvg	-	-	0.71	1x	1x	1x
		Local rand-k	$2e^5$	-	0.75	1x	10x	1.9x
		Fed2Com	$2e^4$	$5e^4$	**0.94**	1.7x	**103x**	**3.3x**
ResNet9	CIFAR10	FedAvg	-	-	0.76	1x	1x	1x
		Local rand-k	$6.5e^5$	-	0.95	1x	10.1x	1.8x
		Fed2Com	$3e^5$	$5e^4$	**0.97**	8.3x	**131.4x**	**15.8x**
DistilBERT	Sentiment140	FedAvg	-	-	0.33	1x	1x	1x
		Local rand-k	$6.6e^6$	-	0.34	1x	10x	1.8x
		Fed2Com	$3e^5$	$5e^4$	**0.74**	3.9x	**1339x**	**7.7x**
GPT2-Small	PersonaChat	FedAvg	-	-	0.79	1x	1x	1x
		Local rand-k	$1.5e^7$	-	0.78	1x	7.8x	2x
		Fed2Com	$5e^5$	$5e^4$	**0.80**	6x	**2334x**	**13.3x**

TABLE II
The accuracy and compression ratio of FedAvg, rand-k, and Fed2Com. A compression ratio of m means that the communication overhead reduces to one in m of the original (FedAvg) communication overhead.

- Can Fed2Com achieve a high compression ratio while maintaining good accuracy?
- How effective is biased compression with error feedback (EF) at the cloud in stabilizing the training process for non-i.i.d cases?
- How does local unbiased compression influence the framework convergence?
- How does the compression ratio influence the model convergence?

A. Experiment Setting

We use four distinct model-dataset combinations that correspond to various problem domains of DNN: a custom CNN with MNIST [18], ResNet9 [15] with CIFAR10 [16], DistilBERT [19] with Sentiment140 [20], and GPT2-Small [21] with PersonaChat [22]. ResNet9 with CIFAR10 has been discussed in §III-A, so here we introduce the other three.

CNN: We build our own CNN model as detailed in Table. I with 2.1M parameters. To create a non-i.i.d version of MNIST, we distribute the dataset to 30,000 clients such that each has 2 images of the same digit.

DistilBERT: DistilBERT is a fast and light transformer model based on BERT with 66 million parameters [19]. We fine-tune the pre-trained DistilBERT on the Sentiment140 dataset [20], which contains 1,600,498 tweets. Each tweet's sentiment has been annotated. This dataset is naturally non-i.i.d with 660,120 users.

GPT2-Small: We fine-tune a small version of OpenAI's GPT2 text generation model [21] on PersonaChat [22], a chit-chat dataset consisting of conversations between Amazon Mechanical Turk workers who were assigned faux personalities to act out. There are a total of 17,568 clients based on the personality that was assigned.

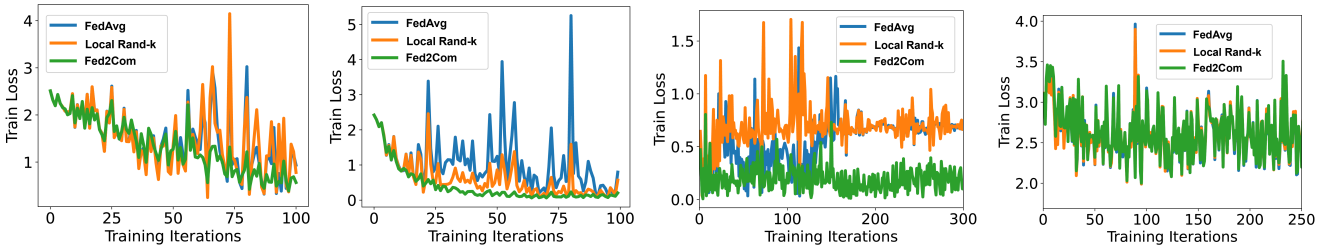
The proposed Fed2Com works with any current compression methods. Here for illustration, the unbiased and biased compressors in Fed2Com are typical rand-k and top-k sparsification. Note that for local updates, we not only perform unbiased compression, i.e., rand-k sparsification, but we also leverage a compact data structure, i.e., count sketch, to achieve dual compression for uplink communication. Meanwhile, the top-k at cloud indicates that we select the top k significant global updates to update the global parameters.

We set the hyperparameters as follows: number of selected clients per round $n = 100$, global learning rate $\gamma = 1$, global momentum $\rho = 0.9$, local batch size $l = 4$ for GPT2-Small and $l = 64$ for DistilBERT, local epoch $K = 4$, and local learning rate $\rho_l = 2e^{-3}$. We run all experiments on four A100-SXM-40G GPUs using PyTorch.

B. Effectiveness of Fed2Com

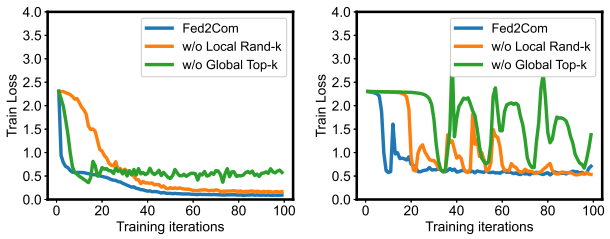
We now present the overall performance of Fed2Com. From Table II, we can see that Fed2Com significantly outperforms competing methods in the regime of very small local datasets and non-i.i.d. data, since Fed2Com consistently reaches the highest compression ratio while retaining the highest validation accuracy for all four models. In particular, we can see severe performance degradation of FedAvg and local rand-k on CIFAR10 and Sentiment140, which can be attributed to the combined effect of unbalanced datasets and local steps. To better illustrate this point, we present the loss curve of the four models in Fig 4. From the loss curves of CNN and ResNet9 trained from scratch (Fig. 4(a) and 4(b)), we see that Fed2Com's loss consistently lies lower than that of the other two methods, demonstrating its better convergence. Meanwhile, the loss curve of Fed2Com is smoother than that of FedAvg and local rand-k, which confirms that our proposed Fed2Com can withstand the influence of the dataset heterogeneity.

DistilBERT and GPT2-Small are pre-trained models fine-tuned on our datasets. We can observe that Fed2Com's loss on DistilBERT is consistently smaller than the other two methods. For GPT2-Small, however, all three methods have very similar loss curves as in Fig. 4(d). The reason is that, clients in PersonaChat are categorized according to the personality they were assigned, yielding a data distribution closer to i.i.d. than other datasets we use. With i.i.d datasets then, Fed2Com achieves comparable accuracy as FedAvg while enjoying much less communication overhead.



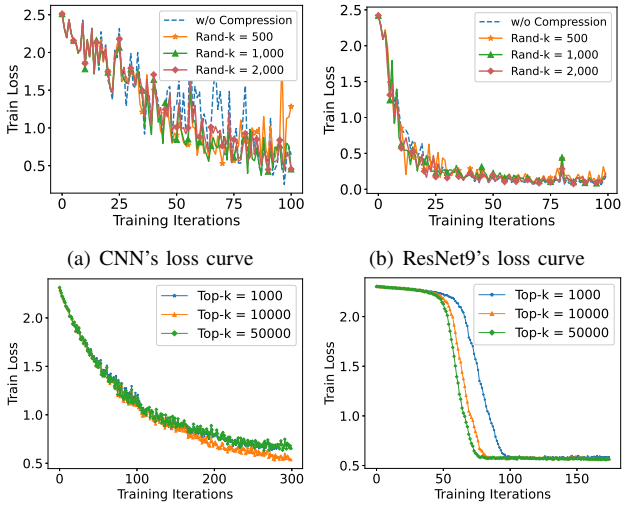
(a) CNN on MNIST (b) ResNet9 on CIFAR10 (c) DistilBERT on Sentiment140 (d) GPT2-small on PersonaChat

Fig. 4. Comparison of the loss convergence of FedAvg, local rand-k and Fed2Com.



(a) CNN's loss curve (b) ResNet9's loss curve

Fig. 5. Training process with different compression methods.



(a) CNN's loss curve (b) ResNet9's loss curve

(c) CNN's loss curve (d) ResNet9's loss curve

Fig. 6. Training process under different compression ratio.

In addition, from Table II we observe that the upload compression ratio is much larger than that of local rand-k, demonstrating the effectiveness of our upload compression method with the combination of rand-k and sketch. Indeed, upload compression is more critical since typically a client's uplink bandwidth is more constrained than downlink.

C. Effectiveness of Biased Compression with Error Feedback

Based on our observation in §III-B, cloud top-k with error correction contributes to the stabilization of the training process since the biased compressor enjoys lower gradient variances while EF compensates for the error. Here we investigate the Fed2Com's performance without top-k at the cloud. From Fig 4, we can see that in three of four models, Fed2Com outperforms local rand-k, demonstrating the necessity of performing top-k with EF at the cloud side. Fig 5 shows the training loss curve with and without different compression methods on the cloud's and clients' sides. To develop fair compression, we keep local sketch in all settings. Here we use CNN and ResNet9 as training models. Each client performs local steps $K = 4$, and the total rounds T is 100. Compared with Fed2Com, the one without global top-k will either can not converge to an optimum, on CNN, or even fail to converge, on ResNet9.

D. Influence of Unbiased Compression

We find in §III-A that rand-k helps to stabilize the training process as its learning curves are smoother than vanilla FedAvg, particularly with non-i.i.d. datasets. In this section, we evaluate and compare the convergence performance on different compression settings.

As demonstrated in Fig. 5, the training loss of ResNet9 model without local rand-k fluctuates dramatically, while the loss value of our Fed2Com drops rapidly at the beginning and is followed by a stable decrease. For the CNN model, the loss curve of Fed2Com falls much faster and smoother than the one without local rand-k. In general, the Fed2Com outperforms others on both converging rate and stability.

E. Influence of Compression Ratio

In this section, we evaluate the impact of the upload compression ratio on convergence performance to thoroughly understand the overall influence of rand-k. The results are shown in Fig. 5.

We first vary the number of uploaded parameters from 500 to 2,000. Observe that albeit merely uploading 500 parameters per iteration, Fed2Com can still converge with relatively high accuracy for both CNN and ResNet9. The upload compression gain reaches 4,130 and 13,000 for two models, respectively. In particular, for CNN, we can see from Fig. 6(a) that the loss curve without local compression oscillates more wildly than that with rand-k compression, demonstrating the necessity of local compression. Yet, the compression ratio cannot be set arbitrarily aggressive as we observe a sudden increase in the

CNN loss at the end of training when only 500 parameters are uploaded. Meanwhile, when we set k to 300, training does not converge for both models. Therefore, an appropriate compression ratio can help stabilize the training process, as long as it is not aggressively small.

Regarding biased compression in the cloud, we conducted experiments by varying the number of parameters broadcasted from 1000 to 50000. The results from Fig. 6(c) show that the convergence process is slower for both small and large k values. This is because a small k value filters out too much useful information, while a large k value cannot effectively remove noise. In the case of ResNet, as show in Fig. 6(d), different top-k values did not affect the convergence result. Higher top-k values allowed for faster reaching of the optimum. This is due to the fact that all top-k values tested remained below the threshold that could negatively impact the training process, since ResNet has significantly more parameters than CNN.

VI. Conclusion

In this paper, we propose Fed2Com, which performs unbiased compression at clients and biased compression with error feedback at the cloud. On the one hand, Fed2Com randomly sparsifies the local updates to reduce uplink communication and withstand the influence of heterogeneous distribution. Since the local compression is unbiased, there is no need to carry out error feedback on the client side. On the other hand, biased compression with error accumulation can be easily performed on the cloud side to stabilize training, and non-participating clients can stay relatively up-to-date with the current model, reducing the number of new parameters that need to be downloaded. The effectiveness of Fed2Com is verified on two image recognition tasks and two language modeling tasks, and the results demonstrate that our proposed Fed2Com outperforms FedAvg and local rand-k on both compression ratio and validation accuracy for all four tasks.

VII. acknowledgement

This work is supported in part by funding from Huawei (CUHK 7010691, 8601677)

References

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

[2] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, "Advances and open problems in federated learning," *Foundations and Trends® in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, 2021.

[3] H. Yang, J. Liu, and E. S. Bentley, "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)*. IEEE, 2021, pp. 1–8.

[4] S. P. Karimireddy, S. Kale, M. Mohri, S. J. Reddi, S. U. Stich, and A. T. Suresh, "Scaffold: Stochastic controlled averaging for on-device federated learning." 2019.

[5] A. K. Sahu, T. Li, M. Sanjabi, M. Zaheer, A. Talwalkar, and V. Smith, "On the convergence of federated optimization in heterogeneous networks," *arXiv preprint arXiv:1812.06127*, vol. 3, p. 3, 2018.

[6] A. Beznosikov, S. Horváth, P. Richtárik, and M. Safaryan, "On biased compression for distributed learning," *arXiv preprint arXiv:2002.12410*, 2020.

[7] J. Wang and G. Joshi, "Cooperative sgd: A unified framework for the design and analysis of communication-efficient sgd algorithms," *arXiv preprint arXiv:1808.07576*, 2018.

[8] W. Luping, W. Wei, and L. Bo, "Cmfl: Mitigating communication overhead for federated learning," in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2019, pp. 954–964.

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

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

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

[12] S. Horváth and P. Richtárik, "A better alternative to error feedback for communication-efficient distributed learning," *arXiv preprint arXiv:2006.11077*, 2020.

[13] Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally, "Deep gradient compression: Reducing the communication bandwidth for distributed training," *arXiv preprint arXiv:1712.01887*, 2017.

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

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

[16] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[17] H. Yu, R. Jin, and S. Yang, "On the linear speedup analysis of communication efficient momentum sgd for distributed non-convex optimization," in *International Conference on Machine Learning*. PMLR, 2019, pp. 7184–7193.

[18] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[19] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," *arXiv preprint arXiv:1910.01108*, 2019.

[20] A. Go, R. Bhayani, and L. Huang, "Twitter sentiment classification using distant supervision," *CS224N project report, Stanford*, vol. 1, no. 12, p. 2009, 2009.

[21] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[22] S. Zhang, E. Dinan, J. Urbanek, A. Szlam, D. Kiela, and J. Weston, "Personalizing dialogue agents: I have a dog, do you have pets too?" *arXiv preprint arXiv:1801.07243*, 2018.