# Communication-efficient Federated Learning with Single-Step Synthetic Features Compressor for Faster Convergence

Yuhao Zhou[1,2], Mingjia Shi[1,2], Yuanxi Li[3], Yanan Sun[1,2], Qing Ye[1,2,*], Jiancheng Lv[1,2,*]
[1] Sichuan University
[2] Engineering Research Center of Machine Learning and Industry Intelligence
[3] University of Illinois at Urbana-Champaign
{sooptq, 3101ihs}@gmail.com, yuanxi3@illinois.edu, {yeqing, ysun, lvjiancheng}@scu.edu.cn

## Abstract

*Reducing communication overhead in federated learning (FL) is challenging but crucial for large-scale distributed privacy-preserving machine learning. While methods utilizing sparsification or other techniques can largely reduce the communication overhead, the convergence rate is also greatly compromised. In this paper, we propose a novel method named Single-Step Synthetic Features Compressor (3SFC) to achieve communication-efficient FL by directly constructing a tiny synthetic dataset containing synthetic features based on raw gradients. Therefore, 3SFC can achieve an extremely low compression rate when the constructed synthetic dataset contains only one data sample. Additionally, the compressing phase of 3SFC utilizes a similarity-based objective function so that it can be optimized with just one step, considerably improving its performance and robustness. To minimize the compressing error, error feedback (EF) is also incorporated into 3SFC. Experiments on multiple datasets and models suggest that 3SFC has significantly better convergence rates compared to competing methods with lower compression rates (i.e., up to 0.02%). Furthermore, ablation studies and visualizations show that 3SFC can carry more information than competing methods for every communication round, further validating its effectiveness.*

## 1. Introduction

Until now, federated learning [22] (FL) is deemed as one of the most promising distributed techniques [8, 3] to tackle the isolated data island problem with privacy guarantees. However, the training process of FL involves frequent exchanging of model parameters between central servers and participating clients, which is becoming increasingly expensive, especially considering the rapid growth of model size today [6, 16, 10]. Moreover, participating clients of FL typically operate at unreliable and limited network connection rates compared to data centers [15], further hindering the large-scale deployments of FL. Consequently, communication is becoming the primary bottleneck for flexible FL at the scale [5].

To explore possible approaches for reducing communication overhead in FL, various methods have been proposed targeting different objectives. The work in [30, 21] applied top-$k$ sparsification to the gradients so that only the most important information is transmitted at each epoch. Moreover, Wangni [33] reported that using top-$k$ sparsification with error feedback (EF), the communication overhead of ResNet-50 [12] trained on ImageNet [24] could be reduced by 99.6% while maintaining nearly the same model accuracy. On the other hand, The work in [2, 4] employed quantification to represent gradients by a lower precision data type with a considerably smaller size. Later Karimireddy [17] introduced error feedback to quantification as well, substantially improving the rate of convergence. In [11], instead of gradients, several data samples distilled from the full training dataset were transmitted as they are much smaller than the gradients, and they can produce similar gradients through back-propagation. More recently, Li [19] and Wu [34] proposed compressing and decompressing communication data using compressed sensing and knowledge distillation, respectively.

While the methods mentioned above have been proven useful in reducing communication overhead reduction, empirical evidence indicates that they both suffer from degraded model convergence rates. This means that the model is expected to converge much slower with a smaller compression rate, as demonstrated in Figure 1. Here, the compression rate is defined as Equation 1. In this paper, we propose a single-step synthetic features compressor (3SFC), to boost the convergence rate during training and carry more information under a limited communication budget. Instead

*Corresponding Author

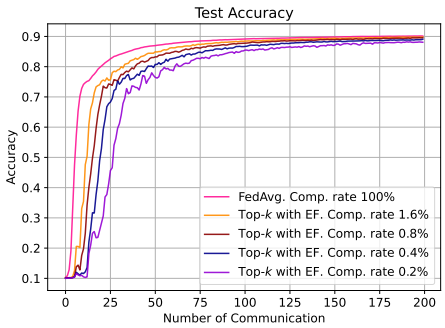



Figure 1: Test accuracy of MLP (199,210 parameters) trained on non-i.i.d. MNIST dataset with 20 clients. The rate of convergence reduces as the compression rate decreases.

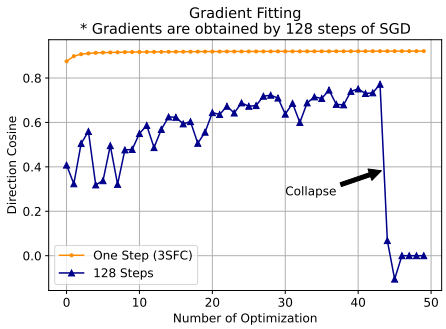



Figure 2: When trying to fit gradients obtained by 128 steps of SGD for 128 steps of simulation using the method in [11], it should be perfectly fitted instead of collapsed. On the other hand, using 1 step of simulation (3SFC) requires less computation and storage but achieves significantly better fitting results.

of transmitting raw gradients directly, 3SFC first constructs a tiny synthetic dataset for the FL model. Then a scaling coefficient is calculated to minimize the compression error. Finally, the constructed synthetic dataset and the scale coefficient are transmitted to the server. In addition, the error feedback [29] is also incorporated into 3SFC to further minimize the overall compression error.

$$\text{Comp. Rate} = \frac{\text{Comp. Size}}{\text{Uncomp. Size}} = \frac{1}{\text{Comp. Ratio}}. \quad (1)$$

Our contributions can be summarized as following:

1. Instead of transmitting gradients employed by most existing compression methods, 3SFC only transmits a tiny set of model inputs and labels, which is independent of the model architecture. Consequently, 3SFC can achieve an extremely low compression rate.

2. A similarity-based objective function is employed to construct synthetic inputs and labels, which drastically lowers the time and space complexity and improves the performance and robustness of 3SFC. Moreover, the error feedback is incorporated into 3SFC to minimize the overall compressing error and therefore boost the convergence rate. These design choices make 3SFC an effective solution for achieveing communication-efficient FL while maintaining model accuracy.

3. 3SFC can achieve a significantly better convergence rate compared to competing methods under the same and even lower communication budget (*i.e.*, up to a compression ratio of 3600×). Ablation study and other visualizations further validate the efficiency of 3SFC against other state-of-the-art works. The code is open-sourced for reproduction [1].

[1] https://github.com/Soptq/iccv23-3sfc

## 2. Related Work

**Sparsification**: Methods in [1, 21, 33] utilized sparsifiers to filter and send partial gradients to greatly reduce the communication overhead. Typical sparsifiers include random-$k$, top-$k$, etc. DGC [21] and STC [26] are considered as current state-of-the-arts. Recently, Sahu [25] formally showed that top-$k$ is the communication-optimal sparsifier under a limited communication budget.

**Quantification**: Methods in [4, 2, 27] replaced the default 32-bit data type in Machine Learning (ML) training with the 8-bit or even 1-bit data type before communicating, and thereby reducing communication overhead. Compared to sparsification methods that can achieve a compression rate of 1/100 or even lower, quantification can at most achieve a compression rate of 1/32.

**Data distillation (Synthetic dataset) for FL**: Recent work [11, 13] have proposed using several synthetic data samples to represent gradients in FL. Since local models in FL are optimized for multiple steps locally, The synthesis process will first generate a synthetic dataset, then simulate optimizing its local model for multiple steps using the synthetic dataset, and finally, utilize the minimization of the $\ell_2$ distance between simulated and real model weights as its objective function to optimize the synthetic dataset. While the multiple steps of simulation is intuitive, empirical results suggest that it leads to not only a high level of time and space complexity for synthesizing (*i.e.*, calculate gradients and store intermediate model weights multiple times), but also great instability and possible collapse, especially for relatively large models and datasets. Figure 2 demonstrates such a collapse. Moreover, gradients of trainable parameters before the collapse are visualized in Figure 3, indicating that a phenomenon similar to the gradient explosion

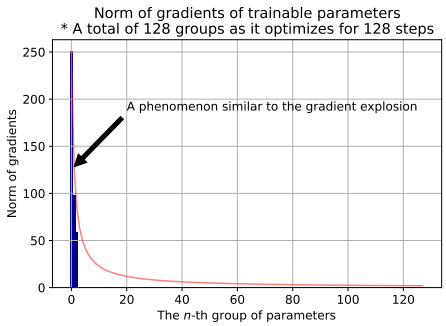



Figure 3: Before the collapse of the method in [11] in Figure 2, the gradients of its trainable parameters exhibit a phenomenon similar to the gradient explosion, where the magnitude of gradients increases as they backpropagate from the 128-th to the first group of parameters. This could be a possible reason for the collapse.

had occurred due to the multiple steps of simulation. Consequently, since previous work hardly converges under our experimental settings involving extremely low compression rate and relatively large models and datasets as Section 5 demonstrated, they will not be compared in our formal experiments.

To alleviate the above-mentioned problems for data distillation to achieve both computation and communication efficient gradient compression, 3SFC employs a similarity-based objective function, instead of $\ell_2$-based objective function, to optimize the synthetic dataset. Empowered by the new objective function, 3SFC optimizes the synthetic dataset only once compared to dozens and hundreds by previous work, substantially enhancing the performance and stability under low compression rates and large models and datasets. Moreover, error feedback is introduced to the data distillation realm for the first time, further helping the optimization converge.

**Others**: Li [19] proposed utilizing compressed sensing for communication data compressing and decompressing. Wu [34] used knowledge distillation to distill the learned knowledge from the local model to a smaller model before communication.

## 3. Problem Formulation

Assume there are $N$ clients participating in a FL training process, where the $i$-th client has a local dataset $D_i$ that obeys distribution $P_i$, and a loss function $F_i(D_i, w_i)$ where $w_i$ is the weight of its model $M_i$. Note that in vanilla FL, all clients and servers share the same model architecture, *i.e.,* $M_1 = M_2 = ... = M_N = M$. The objective of FL is to solve Equation 2:

$$\min_{w \in \mathbb{R}^d} G(F_1(D_1, w), F_2(D_2, w), ..., F_N(D_N, w)), \quad (2)$$

where $G(\cdot)$ is the linear aggregation function satisfying the sum of aggregation weights equals 1. Typical aggregation functions include arithmetic mean and weighted average based on $|D_i|$ [22] where $|\cdot|$ is the size of the $\cdot$. The global model $w$ at the $t$-th communication round is updated by Equation 3:

$$\begin{aligned} w^{t+1} &= G(w_1^t, w_2^t, ..., w_N^t) \\ &= w^t - G(\boldsymbol{g}_1^t, \boldsymbol{g}_2^t, ..., \boldsymbol{g}_N^t)' \end{aligned} \quad (3)$$

where $\boldsymbol{g}_i^t = w^t - w_i^t$ denotes the model weight differences after locally training for $K$ rounds, and can be seen as accumulated gradients. Generally, to reduce communication overhead, a compressor $\mathcal{C}$ is applied to each client's $\boldsymbol{g}_i^t$, so that the global model $w$ can be updated by Equation 4:

$$w^{t+1} = w^t - G(\mathcal{C}(\boldsymbol{g}_1^t), \mathcal{C}(\boldsymbol{g}_2^t), ..., \mathcal{C}(\boldsymbol{g}_N^t)). \quad (4)$$

As a result, the objective of communication compressing can be modeled as Equation 5:

$$\mathcal{C}^* = \arg\min ||\mathcal{C}(\boldsymbol{g}_i^t) - \boldsymbol{g}_i^t||^2 \text{ s.t. } ||\mathcal{C}(\boldsymbol{g}_i^t)||_0 \leq B, \quad (5)$$

where $B$ is the communication budget, constraining the maximum size of communication data at each communication round, and $||\cdot||$ measures the distances of $\cdot$. Moreover, letting $\boldsymbol{\epsilon}_i^t = ||\mathcal{C}(\boldsymbol{g}_i^t) - \boldsymbol{g}_i^t||$ denotes the compression error at time $t$, then the error feedback can be utilized to optimize this error term by adding it to the $\boldsymbol{g}_i^{t+1}$. Thus, with error feedback, the global model can be updated as Equation 6:

$$\begin{cases} w^{t+1} = w^t - G(\mathcal{C}(\boldsymbol{g}_1^t + \boldsymbol{\epsilon}_1^t), \mathcal{C}(\boldsymbol{g}_2^t + \boldsymbol{\epsilon}_2^t), ..., \mathcal{C}(\boldsymbol{g}_N^t + \boldsymbol{\epsilon}_N^t)), \\ \boldsymbol{\epsilon}_i^{t+1} = \boldsymbol{g}_i^t + \boldsymbol{\epsilon}_i^t - \mathcal{C}(\boldsymbol{g}_1^t + \boldsymbol{\epsilon}_1^t). \end{cases} \quad (6)$$

## 4. Our Approach

The general architecture of 3SFC is illustrated in Figure 4. At each epoch, the $i$-th client first trains its local model using its local dataset. After training, accumulated gradients can be obtained by subtracting the global model weights from the latest local model weights. Then, the $i$-th client will utilize the encoder to compress the averaged gradients into a synthetic dataset $D_{syn,i}^t$ that fits the communication budget. When the compressed data is received by the server, the server will first decode the compressed data into accumulated gradients, and then it will aggregate the gradients and update the global model. As seen from Figure 4, in 3SFC, the compressor $\mathcal{C}$ consists of an encoder and a decoder, where the encoder is located on the clients and the decoder is placed on the servers.

### 4.1. Encoder with error feedback

The encoder in 3SFC is responsible for compressing $\boldsymbol{g}_i^t$ into a synthetic dataset $D_{syn,i}^t$ and a scaling coefficient $s_i^t$.

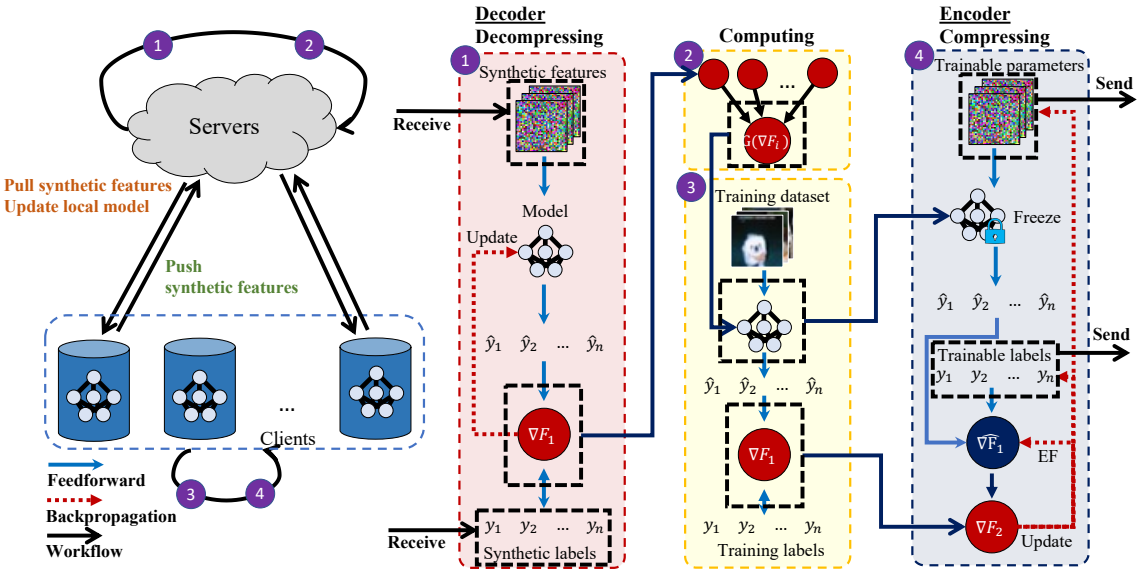



Figure 4: The general architecture of 3SFC. When compressing in ❹, a set of trainable parameters and labels will first be fed into the frozen local model to calculate model gradients. Then, calculated model gradients will be compared with real model gradients to optimize the trainable parameters and labels (i.e., synthetic features). When decompressing in ❶, simply feed the local model with the received synthetic inputs and labels and use the generated gradients to update the global model.

The objective of the encoder can be described by Equation 7:

$$\begin{cases} \min\limits_{D^t_{syn,i}, s^t_i} ||s^t_i \nabla_{w^t} F_i(D^t_{syn,i}, w^t) - \boldsymbol{g_i^t} - \boldsymbol{\epsilon}^t_i||^2 + \lambda {D^t_{syn,i}}^2 \\ \text{s.t. } ||D^t_{syn,i}||_0 + 1 \leq B, \end{cases} \tag{7}$$

where $\boldsymbol{g_i^t}$ denotes the differences between the global model $w^t$ and its latest local model, *i.e.*, $\boldsymbol{g_i^t} = w^t - w^t_i$ for clients. $\lambda {D^t_{syn,i}}^2$ is an $\ell_2$ regularization term to constrain the solution of $D^t_{syn,i}$ for better stability. Note that here the global model $w^t$ is passed into $F_i(\cdot)$ instead of $w^t_i$, because $w^t$ is the initial weight of every client's local optimization process at each epoch. Since $\boldsymbol{g_i^t} + \boldsymbol{\epsilon_i^t}$ is fixed, $s^t_i$ can be derived from $\nabla_{w^t} F_i(D^t_{syn,i}, w^t)$ as shown in Equation 8:

$$\begin{aligned} s^t_i &= \frac{||\boldsymbol{g_i^t} + \boldsymbol{\epsilon}^t_i||}{||\nabla_{w^t} F_i(D^t_{syn,i}, w^t)||} cos(\theta) \\ &= \frac{(\boldsymbol{g_i^t} + \boldsymbol{\epsilon}^t_i) \cdot \nabla_{w^t} F_i(D^t_{syn,i}, w^t)}{||\nabla_{w^t} F_i(D^t_{syn,i}, w^t)||^2}, \end{aligned} \tag{8}$$

where $\theta$ is the angle between two $\boldsymbol{g_i^t} + \boldsymbol{\epsilon}^t_i$ and $\nabla_{w^t} F_i(D^t_{syn,i}, w^t)$. Consequently, the objective described in Equation 7 is equivalent to the following optimization problem:

$$\begin{cases} \min\limits_{D^t_{syn,i}} 1 - |\frac{\nabla_{w^t} F_i(D^t_{syn,i}, w^t) \cdot (\boldsymbol{g_i^t} + \boldsymbol{\epsilon}^t_i)}{||\nabla_{w^t} F_i(D^t_{syn,i}, w^t)|| ||\boldsymbol{g_i^t} + \boldsymbol{\epsilon}^t_i||}| + \lambda {D^t_{syn,i}}^2 \\ \text{s.t. } ||D^t_{syn,i}||_0 + 1 \leq B. \end{cases} \tag{9}$$

Namely, The objective is to find a synthetic dataset $D_{syn}$ that produces gradients that are most similar to $\boldsymbol{g_i^t}$ in terms of the direction. After solving Equation 9, $s^t_i$ can be thus calculated by Equation 8 and the compression error $\boldsymbol{\epsilon}^t_i$ can be updated by Equation 6. Finally, $D^t_{syn,i}$ and $s^t_i$ will be uploaded to others to represent the local gradients of client $i$.

### 4.2. Decoder

After receiving $D^t_{syn,i}$ and $s^t_i$ from others, the decoder at server $j$ will attempt to reconstruct the gradients for local model updating by the following equation:

$$\boldsymbol{g_i^t} + \boldsymbol{\epsilon}^t_i = s^t_i \nabla_{w^t} F_j(D^t_{syn,i}, w^t). \tag{10}$$

Note that the success of the reconstruction depends on the assumption that the server $j$ has access to the global model $w^t$ and $F_i(\cdot) = F_j(\cdot)$, which can be easily satisfied. Finally, for servers, following Equation 6, the global model of server $j$ can be updated accordingly.

### 4.3. Algorithm and complexity analysis

The pseudocode of 3SFC is presented in Algorithm 1. In 3SFC, during the training process, clients will solve two op-

**Algorithm 1** 3SFC(One Epoch)

**Input**: global model $w^t$, local dataset $D_i$, learning rate $\eta_i$, accumulated gradient $\boldsymbol{\epsilon}_i^t$, regularization parameter $\lambda$
**Parameter**: communication budget $B$, number of global epoch $E$, number of local iteration $K$, number of 3SFC iteration $S$, number of clients $N$, aggregation function $G$
**Output**: global model $w^{t+1}$
**Clients:**

1: **for** each client $i$ from 1 to $N$ **in parallel do**
2: initialize $D_{syn,i}^t$ where $||D_{syn,i}^t||_0 + 1 \leq B$
3: **for** each local iteration $e$ from 1 to $K$ **do**
4: $w_i^t = w_i^t - \eta_i \nabla_{w_i^t} F_i(D_i, w_i^t)$
5: **end for**
6: $\boldsymbol{g}_i^t = w_i^t - w_i$
7: **for** each $s$ from 1 to $S$ **do**
8: $D_{syn,i}^t = D_{syn,i}^t - \eta_i \nabla_{D_{syn,i}^t}(1 - |\frac{\nabla_{w^t} F_i(D_{syn,i}^t, w^t) \cdot (\boldsymbol{g}_i^t + \boldsymbol{\epsilon}_i^t)}{||\nabla_{w^t} F_i(D_{syn,i}^t, w^t)|| ||\boldsymbol{g}_i^t + \boldsymbol{\epsilon}_i^t||}| + \lambda {D_{syn,i}^t}^2)$
9: **end for**
10: $s_i^t = \frac{(\boldsymbol{g}_i^t + \boldsymbol{\epsilon}_i^t) \cdot \nabla_{w^t} F_i(D_{syn,i}^t, w^t)}{||\nabla_{w^t} F_i(D_{syn,i}^t, w^t)||^2}$
11: $\boldsymbol{\epsilon}_i^{t+1} = \boldsymbol{\epsilon}_i^t + \boldsymbol{g}_i^t - \nabla_{w^t} F_i(D_{syn,i}^t, w^t)$
12: **return** $D_{syn,i}^t$, $s_i^t$, $\boldsymbol{\epsilon}_i^{t+1}$
13: **end for**

**Servers:**

1: **for** each client $i$ from 1 to $N$ **do**
2: receive $D_{syn,i}^t$, $s_i^t$
3: $\boldsymbol{g}_i^t + \boldsymbol{\epsilon}_i^t = s_i^t \nabla_{w^t} F_i(D_{syn,i}^t, w^t)$
4: **end for**
5: $w^{t+1} = w^t - G(\boldsymbol{g}_1^t + \boldsymbol{\epsilon}_1^t, \boldsymbol{g}_2^t + \boldsymbol{\epsilon}_2^t, ..., \boldsymbol{g}_N^t + \boldsymbol{\epsilon}_N^t)$
6: **return** $w^{t+1}$

timization problems instead of one compared to the vanilla FL method FedAvg [22]: the empirical risk minimization problem on the local dataset (Line 4) and Equation 9 for compression (Line 8). The solvers to these two problems are not nested, meaning the time complexity of 3SFC equals $\mathcal{O}(NE(K+S))$. In terms of the space complexity, 3SFC additionally stores the $w^t$, $D_{syn,i}^t$, $s_i^t$ and $\boldsymbol{\epsilon}_i^t$, which are all fixed size parameters. Hence, 3SFC shares the same space complexity, $\mathcal{O}(N)$, with FedAvg as well.

## 5. Experiments

**Datasets:** Following the conventions of the community [26, 36, 4], five datasets including MNIST [9], FMNIST [35], EMNIST [7], Cifar10 and Cifar100 [18] are used in the experiments. To simulate the Non-i.i.d. characteristic, all datasets are manually partitioned into multiple subsets based on the Dirichlet distribution, which is commonly used in the FL setting [31, 20]. Figure 5 illustrates our partitions. As can be seen, different clients own different datasets in terms of both quantity and category.

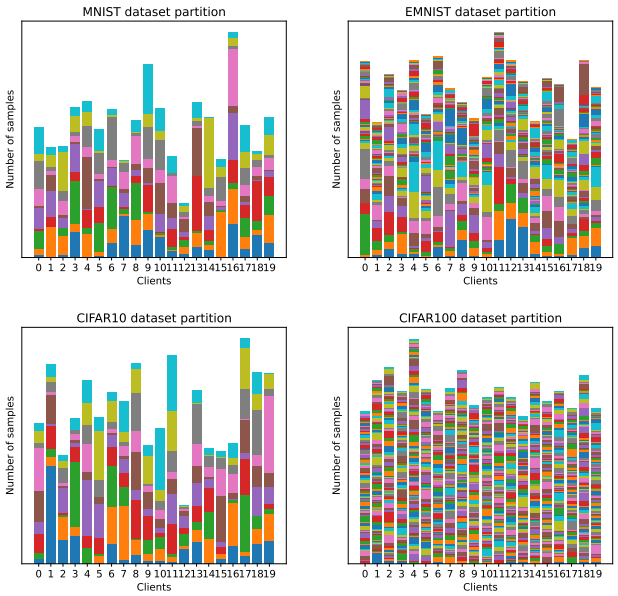



Figure 5: Illustration of our manual dataset partitions for 20 clients based on the Dirichlet distribution. Each bar represent a client, and different segments with different colors of a bar represents different labels. As can be seen, different clients have different dataset sizes and dataset distributions, and some clients only have some of the labels.

| Dataset+Model | FedAvg (1×) | FedSynth 1× | FedSynth 250× |
|---|---|---|---|
| MNIST+MLP | 0.9017 | 0.9017 | 0.1359 |
| EMNIST+MLP | 0.6108 | 0.6108 | 0.0192 |
| FMNIST+MLP | 0.8183 | 0.8183 | 0.1216 |
| FMNIST+Mnistnet | 0.8573 | 0.8573 | 0.1318 |

Table 1: Test accuracies of FedSynth in our preliminary experiments with 10 clients after 200 epochs of training. As can be seen, the model is barely optimized (as discussed in Section 2) with FedSynth with an extremely high compression ratio, while other methods like 3SFC and DGC achieve much higher performances as Table 2 illustrated. Consequently, FedSynth is not compared with 3SFC in the latter experiments. These results validate our observations described in Section 2.

**Models:** To cover both simple and complicated learning problems, five models including Multi-Layer Perceptron (MLP), MnistNet, ConvNet, ResNet [12] and RegNet [23] are used in the experiments. Here, MnistNet has two convolutional layers and two linear layers, and ConvNet has four convolutional layers and one linear layer. Additionally, for ResNet and RegNet, all batch normalization layers [14] and dropout layers [28] are deleted from the model as their parameters are not trainable [32]. This simplification has also been used in previous studies[26, 36].

| Methods | MNIST | EMNIST | FMNIST | | Cifar10 | | | Cifar100 | |
|---|---|---|---|---|---|---|---|---|---|
| | MLP | MLP | MLP | Mnistnet | ConvNet | ResNet | RegNet | ResNet | RegNet |
| 10 Clients | | | | | | | | | |
| FedAvg | 0.9017 (1.0×) | 0.6108 (1.0×) | 0.8183 (1.0×) | 0.8573 (1.0×) | 0.6153 (1.0×) | 0.4759 (1.0×) | 0.4498 (1.0×) | 0.1575 (1.0×) | 0.114 (1.0×) |
| DGC | 0.8663 (250.0×) | 0.5287 (250.0×) | 0.7718 (250.0×) | 0.8065 (1333.3×) | 0.6151 (10.4×) | 0.2113 (3571.4×) | 0.3050 (757.6×) | 0.0138 (3571.4×) | 0.0322 (757.6×) |
| signSGD | 0.8692 (32.0×) | 0.5415 (32.0×) | 0.7550 (32.0×) | 0.8198 (32.0×) | 0.6180 (32.0×) | 0.3687 (32.0×) | 0.2759 (32.0×) | 0.0178 (32.0×) | 0.0391 (32.0×) |
| STC | 0.8848 (32.0×) | 0.5258 (32.0×) | **0.8016** (32.0×) | **0.8427** (32.0×) | **0.6187** (**32.0**×) | **0.4009** (32.0×) | 0.3568 (32.0×) | 0.0088 (32.0×) | 0.0416 (32.0×) |
| 3SFC | **0.8876** (**250.0**×) | **0.5494** (**250.0**×) | 0.7881 (**250.0**×) | 0.8179 (**1333.3**×) | 0.6182 (10.4×) | 0.2567 (**3571.4**×) | **0.3753** (**757.6.0**×) | **0.0466** (**3571.4**×) | **0.0711** (**757.6**×) |
| 20 Clients | | | | | | | | | |
| FedAvg | 0.9013 (1.0×) | 0.6086 (1.0×) | 0.8173 (1.0×) | 0.8572 (1.0×) | 0.6146 (1.0×) | 0.4701 (1.0×) | 0.4646 (1.0×) | 0.1785 (1.0×) | 0.1194 (1.0×) |
| DGC | 0.8808 (250.0×) | 0.5332 (250.0×) | 0.7768 (250.0×) | 0.8207 (1333.3×) | 0.6115 (10.4×) | 0.2542 (3571.4×) | 0.3204 (757.6×) | 0.0101 (3571.4×) | 0.0501 (757.6×) |
| signSGD | 0.8689 (32.0×) | 0.5483 (32.0×) | 0.7522 (32.0×) | 0.8102 (32.0×) | 0.6099 (32.0×) | 0.3673 (32.0×) | 0.3020 (32.0×) | **0.0824** (32.0×) | 0.051 (32.0×) |
| STC | 0.8889 (32.0×) | 0.5512 (32.0×) | **0.8020** (32.0×) | 0.8198 (32.0×) | **0.6125** (**32.0**×) | **0.4111** (32.0×) | 0.3748 (32.0×) | 0.0734 (32.0×) | 0.0499 (32.0×) |
| 3SFC | **0.8918** (**250.0**×) | **0.5556** (**250.0**×) | 0.8013 (**250.0**×) | **0.8217** (**1333.3**×) | 0.6044 (10.4×) | 0.3049 (**3571.4**×) | **0.3854** (**757.6**×) | 0.0532 (**3571.4**×) | **0.0764** (**757.6**×) |
| 40 Clients | | | | | | | | | |
| FedAvg | 0.9003 (1.0×) | 0.6138 (1.0×) | 0.8162 (1.0×) | 0.8559 (1.0×) | 0.6036 (1.0×) | 0.4653 (1.0×) | 0.4597 (1.0×) | 0.0168 (1.0×) | 0.1047 (1.0×) |
| DGC | 0.8775 (250.0×) | 0.5425 (250.0×) | 0.7645 (250.0×) | 0.8297 (1333.3×) | 0.6056 (10.4×) | 0.2807 (3571.4×) | 0.3379 (757.6×) | 0.0094 (3571.4×) | 0.0448 (757.6×) |
| signSGD | 0.8698 (32.0×) | 0.5583 (32.0×) | 0.7546 (32.0×) | 0.8124 (32.0×) | 0.6102 (32.0×) | 0.3779 (32.0×) | 0.3012 (32.0×) | 0.0812 (32.0×) | 0.0475 (32.0×) |
| STC | 0.8886 (32.0×) | **0.5607** (32.0×) | **0.7996** (32.0×) | **0.8310** (32.0×) | 0.6024 (**32.0**×) | **0.4128** (32.0×) | 0.3603 (32.0×) | **0.0818** (32.0×) | 0.0414 (32.0×) |
| 3SFC | **0.8886** (**250.0**×) | 0.5595 (**250.0**×) | 0.7945 (**250.0**×) | 0.827 (**1333.3**×) | **0.6145** (10.4×) | 0.2869 (**3571.4**×) | **0.3835** (**757.6**×) | 0.0560 (**3571.4**×) | **0.0618** (**757.6**×) |

Table 2: Comparison of test accuracy and compression ratio. Note that 3SFC and DGC have much higher compression ratios compared to signSGD and STC due to the limitation of quantification-based methods and the high compressing efficiency of 3SFC and DGC. Consequently, while STC seems to perform well, 3SFC achieves competing or better performance with a significantly lower communication budget. A dedicated comparison of 3SFC and STC is illustrated later to demonstrate the superiority of 3SFC in Section 6.2.

**Competetors:** We compare 3SFC with 4 other methods: FedAvg [22], DGC [21], signSGD with EF [4] and STC [26]. Specifically, FedAvg is a traditional FL training method without any compression, DGC is considered as a state-of-the-art in sparsification, signSGD is a typical quantification method and STC combines sparsification and quantification (*i.e.*, STC sparsifies top-$k$ parameters and quantifies the others). Note that previous work in the data distillation for FL realm (*e.g.*, FedSynth [13] is considered as a state-of-the-art in data distillation for FL realm to achieve communication efficient FL) is not compared in our experiments, as it hardly converges due to the instability and collapse described in Section 2 with high compression ratio and large datasets and models, as Table 1 illustrated. All later experiments are evaluated on a simulated 40 clients cluster. The CUDA version is 11.4, the Python version is 3.9.15 and the PyTorch version is 1.13.0.

# 6. Analysis

## 6.1. Performance comparisons

We first compare the final accuracy of 3SFC with other competing methods after 200 epochs of training. The learning rate is set to 0.01, the batch size is set to 256, local iteration $K$ is set to 5 and $\lambda$ is set to 0 for no regularization. In terms of the compression rate, we set DGC to be the same as 3SFC for all experiments for fair comparisons, because DGC is a sparsification-based method that can have an extremely low compression rate. For quantification-based methods like signSGD and STC, we leave their compression rate to be 1/32, and will later do dedicated evaluations between them and 3SFC.

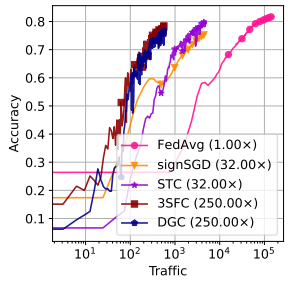
The comprehensive accuracy comparison results of 3SFC and other methods are shown in Table 2. It can be observed that under the same compression rate, 3SFC yields higher test accuracy consistently compared to DGC after training, suggesting that 3SFC brings a faster convergence rate to the model training when the communication budget is limited. On the other hand, 3SFC still achieves comparable model performance compared to signSGD and STC, where the latter two methods communicate much more (*i.e.*, 100× more for ResNet). Figure 6 further validates the effectiveness of 3SFC by visualizing the test accuracy and training loss.

## 6.2. Further comparisons between 3SFC and STC

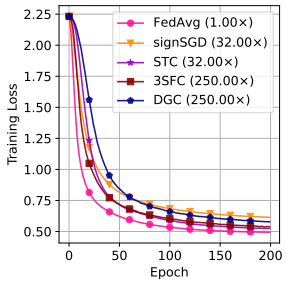
To further evaluate 3SFC compared to STC for fairness, we gradually increase the communication budget of 3SFC and compare both their compression ratio and test accuracy, which is shown in Table 3. As the table suggests, 3SFC can achieve comparable or even better test accuracy while saving a significant amount of communication traffic. For example, when training ResNet on Cifar10 with 10 clients, 3SFC reports a comparable final accuracy (0.3954 compared to 0.4009) while communicating 189.4× fewer data. On the other hand, 3SFC reaches considerably better performance for RegNet on Cifar100 (0.0946 compared to 0.0416) with 384.6× better compression ratio.

## 6.3. Compression efficiency

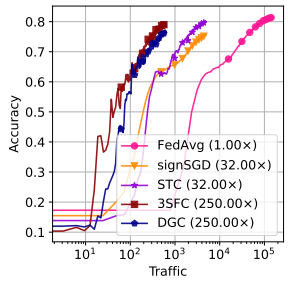
To study why 3SFC achieves a faster convergence rate, we restrain the compression rate of 3SFC and DGC to be



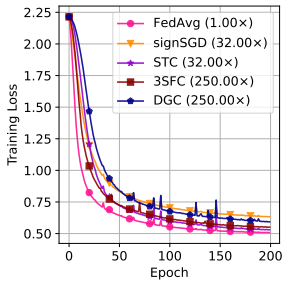

(a) MLP trained on MNIST.



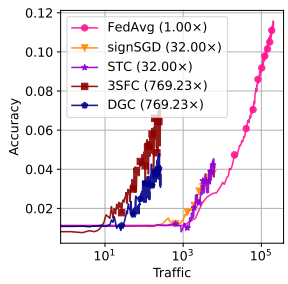

(b) MLP trained on MNIST.



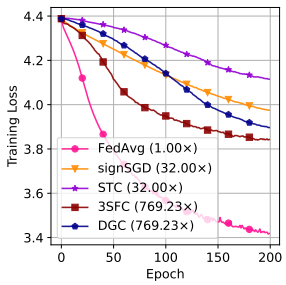

(c) MLP trained on FMNIST.




(d) MLP trained on FMNIST.




(e) RegNet trained on Cifar100.




(f) RegNet trained on Cifar100.

Figure 6: Test accuracy and training loss comparisons after 200 epochs of training. Compared to other methods, 3SFC owns the fastest convergence rate with respect to the amount of traffic communicated, with the highest compression ratio.

the same, and visualize the compression efficiency of 3SFC, DGC, and FedAvg. Here, the compression efficiency stands for how much information the compressed data carry compared to the uncompressed data. Intuitively, the compression efficiency can be represented by the $\ell_2$ distance between compressed and uncompressed data. However, since the compressed data in both 3SFC and DGC are vertical to the uncompressed data (which is illustrated by E quation 8), in this subsection, we will use the cosine similarity between the compressed and uncompressed data as the compression efficiency. The visualization is shown in Figure 7.

In Figure 7, FedAvg has a constant compression efficiency of 1.0, as FedAvg does not compress the data at all. Hence, FedAvg is served as a reference here. Meanwhile,

| Dataset+Model | STC | 3SFC ($2 \times B$) | 3SFC ($4 \times B$) |
|---|---|---|---|
| | | 10 Clients | |
| MNIST+MLP | 0.8848 (32.0×) | **0.8961** (**125.0**×) | 0.8958 (62.5×) |
| EMNIST+MLP | 0.5258 (32.0×) | 0.5820 (**125.0**×) | **0.5955** (62.5×) |
| FMNIST+MLP | 0.8016 (32.0×) | 0.8031 (**125.0**×) | **0.8063** (62.5×) |
| FMNIST+Mnistnet | 0.8427 (32.0×) | 0.8356 (**666.7**×) | **0.8430** (333.3×) |
| Cifar10+Resnet | **0.4009** (32.0×) | 0.3642 (**1785.7**×) | 0.3954 (892.9×) |
| Cifar10+Regnet | 0.3568 (32.0×) | 0.4335 (**378.8**×) | **0.4341** (189.4×) |
| Cifar100+ResNet | 0.0088 (32.0×) | 0.0881 (**1785.7**×) | **0.0989** (892.9×) |
| Cifar100+RegNet | 0.0416 (32.0×) | 0.0946 (**384.6**×) | **0.0952** (192.3×) |
| Avg | 0.4829 (32.0×) | 0.5121 (672.1×) | 0.5205 (336.1×) |
| Std | 0.3234 | 0.3008 | 0.2987 |
| | | 20 Clients | |
| MNIST+MLP | 0.8889 (32.0×) | 0.8948 (**125.0**×) | **0.8963** (62.5×) |
| EMNIST+MLP | 0.5512 (32.0×) | 0.5832 (**125.0**×) | **0.5961** (62.5×) |
| FMNIST+MLP | 0.8020 (32.0×) | 0.8053 (**125.0**×) | **0.8070** (62.5×) |
| FMNIST+Mnistnet | 0.8198 (32.0×) | 0.8343 (**666.7**×) | **0.8372** (333.3×) |
| Cifar10+Resnet | **0.4111** (32.0×) | 0.3450 (**1785.7**×) | 0.3654 (892.9×) |
| Cifar10+Regnet | 0.3748 (32.0×) | 0.4376 (**378.8**×) | **0.4508** (189.4×) |
| Cifar100+ResNet | 0.0734 (32.0×) | 0.0973 (**1785.7**×) | **0.1118** (892.9×) |
| Cifar100+RegNet | 0.0499 (32.0×) | 0.0977 (**384.6**×) | **0.1031** (192.3×) |
| Avg | 0.4964 (32.0×) | 0.5119 (672.1×) | 0.5210 (336.1×) |
| Std | 0.3073 | 0.2997 | 0.2957 |
| | | 40 Clients | |
| MNIST+MLP | 0.8886 (32.0×) | 0.8932 (**125.0**×) | **0.8949** (62.5×) |
| EMNIST+MLP | 0.5607 (32.0×) | 0.5876 (**125.0**×) | **0.5995** (62.5×) |
| FMNIST+MLP | 0.7996 (32.0×) | 0.8027 (**125.0**×) | **0.8073** (62.5×) |
| FMNIST+Mnistnet | 0.8310 (32.0×) | 0.8374 (**666.7**×) | **0.8412** (333.3×) |
| Cifar10+Resnet | **0.4128** (32.0×) | 0.3747 (**1785.7**×) | 0.3695 (892.9×) |
| Cifar10+Regnet | 0.3603 (32.0×) | 0.4481 (**378.8**×) | **0.4503** (189.4×) |
| Cifar100+ResNet | 0.0818 (32.0×) | 0.1041 (**1785.7**×) | **0.1189** (892.9×) |
| Cifar100+RegNet | 0.0414 (32.0×) | 0.0799 (**384.6**×) | **0.0889** (192.3×) |
| Avg | 0.4970 (32.0×) | 0.5160 (672.1×) | 0.5213 (336.1×) |
| Std | 0.3095 | 0.2994 | 0.2972 |

Table 3: Test accuracy and compression ratio comparisons of STC and 3SFC with different communication budgets. 3SFC mostly achieves higher test accuracy while having a higher compression ratio, suggesting 3SFC compresses and decompresses the communication data more efficiently.

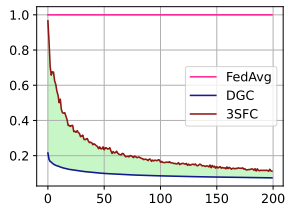
it is clear from the figure that with the same compression rate, 3SFC achieves higher compression efficiency for every communication round (*i.e.*, the green area), meaning that the compression error of 3SFC is lower at every update step of the global model, contributing to the faster convergence rate of the training. Moreover, as error feedback is incorporated into both DGC and 3SFC, the compression error of each communication round will be accumulated into $\mathbf{g}_i^t$ forever. Consequently, the compression efficiency for both DGC and 3SFC decreases gradually as the training progresses.

### 6.4. Ablation study

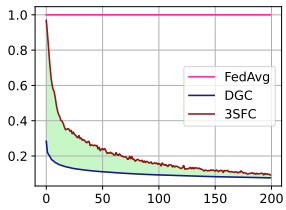
Table 4 shows the ablation study of 3SFC in terms of EF, communication budget $B$ and local iteration $K$. As observed from Table 4, compared to 3SFC w/ EF, disabling EF in 3SFC drastically degrades the model performance after training in all experiments, validating the effectiveness of EF. Moreover, MLP trained on MNIST by 3SFC w/o EF ob-

| Methods | MNIST | EMNIST | FMNIST | | Cifar10 | | | Cifar100 | | Statistics | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | MLP | MLP | MLP | Mnistnet | ConvNet | ResNet | RegNet | ResNet | RegNet | Avg | Std |
| 10 Clients | | | | | | | | | | | |
| 3SFC w/ EF | 0.8876 | 0.5494 | 0.7881 | 0.8179 | 0.6182 | 0.2567 | 0.3753 | 0.0466 | 0.0711 | 0.4901 | 0.3008 |
| 3SFC w/o EF | 0.4580 | 0.2397 | 0.5746 | 0.7324 | 0.4495 | 0.2313 | 0.2559 | 0.0170 | 0.0235 | 0.3313 | 0.2280 |
| 3SFC w/ EF ($2 \times B$) | 0.8961 | 0.5820 | 0.8031 | 0.8356 | 0.6308 | 0.3642 | 0.4335 | 0.0881 | 0.0946 | 0.5253 | 0.2860 |
| 3SFC w/ EF ($4 \times B$) | 0.8958 | 0.5955 | 0.8063 | 0.8430 | 0.6241 | 0.3954 | 0.4341 | 0.0989 | 0.0952 | 0.5320 | 0.2835 |
| 3SFC w/ EF ($K = 1$) | 0.6939 | 0.3152 | 0.6500 | 0.7807 | 0.5207 | 0.2001 | 0.2871 | 0.0104 | 0.0366 | 0.3883 | 0.2689 |
| 3SFC w/ EF ($K = 10$) | 0.8961 | 0.6075 | 0.8212 | 0.8383 | 0.6333 | 0.3099 | 0.3908 | 0.0494 | 0.0778 | 0.5138 | 0.3040 |
| 20 Clients | | | | | | | | | | | |
| 3SFC w/ EF | 0.8918 | 0.5556 | 0.8013 | 0.8217 | 0.6044 | 0.3049 | 0.3854 | 0.0532 | 0.0764 | 0.4994 | 0.2966 |
| 3SFC w/o EF | 0.6707 | 0.1970 | 0.6008 | 0.7450 | 0.4625 | 0.2373 | 0.3062 | 0.0231 | 0.0295 | 0.3635 | 0.2539 |
| 3SFC w/ EF ($2 \times B$) | 0.8948 | 0.5832 | 0.8053 | 0.8343 | 0.6165 | 0.3450 | 0.4376 | 0.0973 | 0.0977 | 0.5235 | 0.2845 |
| 3SFC w/ EF ($4 \times B$) | 0.8963 | 0.5961 | 0.8070 | 0.8372 | 0.6187 | 0.3654 | 0.4508 | 0.1118 | 0.1031 | 0.5318 | 0.2805 |
| 3SFC w/ EF ($K = 1$) | 0.7504 | 0.3387 | 0.6441 | 0.7706 | 0.5249 | 0.2155 | 0.3072 | 0.0126 | 0.0412 | 0.4006 | 0.2709 |
| 3SFC w/ EF ($K = 10$) | 0.9063 | 0.6127 | 0.8289 | 0.8294 | 0.6049 | 0.3095 | 0.4150 | 0.0483 | 0.0831 | 0.5153 | 0.3027 |
| 40 Clients | | | | | | | | | | | |
| 3SFC w/ EF | 0.8886 | 0.5595 | 0.7945 | 0.8270 | 0.6145 | 0.2869 | 0.3835 | 0.0560 | 0.0618 | 0.4969 | 0.2999 |
| 3SFC w/o EF | 0.4830 | 0.2512 | 0.6349 | 0.7575 | 0.4742 | 0.2300 | 0.2917 | 0.0133 | 0.0277 | 0.3515 | 0.2416 |
| 3SFC w/ EF ($2 \times B$) | 0.8932 | 0.5876 | 0.8027 | 0.8374 | 0.6132 | 0.3747 | 0.4481 | 0.1041 | 0.0799 | 0.5268 | 0.2840 |
| 3SFC w/ EF ($4 \times B$) | 0.8949 | 0.5995 | 0.8073 | 0.8412 | 0.6115 | 0.3695 | 0.4503 | 0.1189 | 0.0889 | 0.5313 | 0.2817 |
| 3SFC w/ EF ($K = 1$) | 0.6956 | 0.3129 | 0.6547 | 0.7752 | 0.5243 | 0.2254 | 0.3024 | 0.0127 | 0.0317 | 0.3928 | 0.2669 |
| 3SFC w/ EF ($K = 10$) | 0.9072 | 0.6074 | 0.8277 | 0.8399 | 0.5875 | 0.3099 | 0.4176 | 0.0476 | 0.0748 | 0.5133 | 0.3045 |

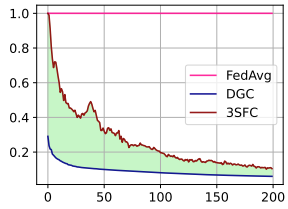
Table 4: The ablation study with different parameters of 3SFC (*i.e.*, with/without EF, communication budgets $B$, local iteration $K$). The configuration for the Base is $1 \times B$ and $K$ = 5. From the table, it is clear that enabling EF in 3SFC has an important role in helping models converge, validating its effectiveness. Moreover, Increasing $B$ or $K$ can both further boost the convergence rate of the training.



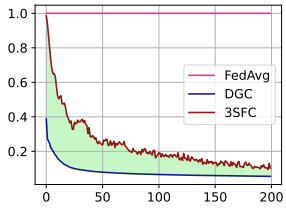

(a) RegNet trained on Cifar10.




(b) RegNet trained on Cifar100.




(c) ResNet trained on Cifar10.




(d) ResNet trained on Cifar100.

Figure 7: Compression efficiency comparisons. 3SFC owns significantly higher compression efficiency compared to DGC under the same compression rate, suggesting 3SFC is a much more efficient compressor compared to DGC.

tained a final test accuracy of 0.4580 with 10 clients (0.8876 for 3SFC w/ EF), 0.6707 with 20 clients (0.8918 for 3SFC w/ EF) and 0.4830 with 40 clients (0.8886 for 3SFC w/ EF). Such a huge performance difference in such a simple learning task effectively suggests that disabling EF also brings more instability and uncertainty to the training process.

In terms of $B$ and $K$, when increasing $B$, the test accuracy of the model increases as well, as more data are being transferred at each communication round. On the other hand, by decreasing the local iteration $K$ from 5 to 1, the test accuracy is reduced significantly since the model has been optimized much less. Contrarily, the test accuracy boosts up when $K$ is set to 10. Consequently, increasing the communication budget $B$ is the most significant way to further boost the convergence rate of the model using 3SFC. For example, by increasing the compression rate from 0.028% ($1 \times B$) to 0.056% ($2 \times B$) for ResNet trained on Cifar100 with 40 clients, the convergence rate of the model gets doubled and the test accuracy after 200 epochs of training also increases from 0.0560 to 0.1041. However, when the communication budget is strictly limited, the convergence rate of 3SFC can be improved as well by setting a larger local iteration $K$.

## 7. Conclusion

In this paper, we propose a single-step synthetic features compressor (3SFC) for communication-efficient FL. 3SFC compresses the data using a similarity-based objective function in a single step, thus saving both compute and storage resources, and maintaining the robustness of the algorithm. Moreover, error feedback is employed to further minimize the compression error. Comparisons of test accuracy and compression ratio show that 3SFC achieves significantly faster convergence rates with lower compression rates. An ablation study demonstrates the role of different parameters, and visualizations of compression efficiency further validates the effectiveness of 3SFC.

## 8. Acknowledgments

This work is supported by the Key Program of National Science Foundation of China (Grant No. 61836006).

## References

[1] Alham Fikri Aji and Kenneth Heafield. Sparse communication for distributed gradient descent. *arXiv preprint arXiv:1704.05021*, 2017. 2

[2] Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. Qsgd: Communication-efficient sgd via gradient quantization and encoding. *Advances in neural information processing systems*, 30, 2017. 1, 2

[3] Tal Ben-Nun and Torsten Hoefler. Demystifying parallel and distributed deep learning: An in-depth concurrency analysis. *ACM Computing Surveys (CSUR)*, 52(4):1–43, 2019. 1

[4] Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Animashree Anandkumar. signsgd: Compressed optimisation for non-convex problems. In *International Conference on Machine Learning*, pages 560–569. PMLR, 2018. 1, 2, 5, 6

[5] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečnỳ, Stefano Mazzocchi, Brendan McMahan, et al. Towards federated learning at scale: System design. *Proceedings of Machine Learning and Systems*, 1:374–388, 2019. 1

[6] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020. 1

[7] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. Emnist: Extending mnist to handwritten letters. In *2017 international joint conference on neural networks (IJCNN)*, pages 2921–2926. IEEE, 2017. 5

[8] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Marc'aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, et al. Large scale distributed deep networks. *Advances in neural information processing systems*, 25, 2012. 1

[9] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012. 5

[10] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity, 2021. 1

[11] Jack Goetz and Ambuj Tewari. Federated learning via synthetic data. *arXiv preprint arXiv:2008.04489*, 2020. 1, 2, 3

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1, 5

[13] Shengyuan Hu, Jack Goetz, Kshitiz Malik, Hongyuan Zhan, Zhe Liu, and Yue Liu. Fedsynth: Gradient compression via synthetic data in federated learning. *arXiv preprint arXiv:2204.01273*, 2022. 2, 6

[14] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015. 5

[15] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021. 1

[16] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020. 1

[17] Sai Praneeth Karimireddy, Quentin Rebjock, Sebastian Stich, and Martin Jaggi. Error feedback fixes signsgd and other gradient compression schemes. In *International Conference on Machine Learning*, pages 3252–3261. PMLR, 2019. 1

[18] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 5

[19] Chengxi Li, Gang Li, and Pramod K Varshney. Communication-efficient federated learning based on compressed sensing. *IEEE Internet of Things Journal*, 8(20):15531–15541, 2021. 1, 3

[20] Qinbin Li, Yiqun Diao, Quan Chen, and Bingsheng He. Federated learning on non-iid data silos: An experimental study. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, pages 965–978. IEEE, 2022. 5

[21] Yujun Lin, Song Han, Huizi Mao, Yu Wang, and William J Dally. Deep gradient compression: Reducing the communication bandwidth for distributed training. *arXiv preprint arXiv:1712.01887*, 2017. 1, 2, 6

[22] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017. 1, 3, 5, 6

[23] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10428–10436, 2020. 5

[24] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015. 1

[25] Atal Sahu, Aritra Dutta, Ahmed M Abdelmoniem, Trambak Banerjee, Marco Canini, and Panos Kalnis. Rethinking gradient sparsification as total error minimization. *Advances in Neural Information Processing Systems*, 34:8133–8146, 2021. 2

[26] Felix Sattler, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek. Robust and communication-efficient federated learning from non-iid data. *IEEE transactions on neural networks and learning systems*, 31(9):3400–3413, 2019. 2, 5, 6

[27] Frank Seide, Hao Fu, Jasha Droppo, Gang Li, and Dong Yu. 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech dnns. In *Fifteenth annual conference of the international speech communication association*, 2014. 2

[28] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014. 5

[29] Sebastian U Stich, Jean-Baptiste Cordonnier, and Martin Jaggi. Sparsified sgd with memory. *Advances in Neural Information Processing Systems*, 31, 2018. 2

[30] Nikko Strom. Scalable distributed dnn training using commodity gpu cloud computing. In *Sixteenth annual conference of the international speech communication association*, 2015. 1

[31] Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H Vincent Poor. Tackling the objective inconsistency problem in heterogeneous federated optimization. *Advances in neural information processing systems*, 33:7611–7623, 2020. 5

[32] Yanmeng Wang, Qingjiang Shi, and Tsung-Hui Chang. Why batch normalization damage federated learning on non-iid data? *arXiv preprint arXiv:2301.02982*, 2023. 5

[33] Jianqiao Wangni, Jialei Wang, Ji Liu, and Tong Zhang. Gradient sparsification for communication-efficient distributed optimization. *Advances in Neural Information Processing Systems*, 31, 2018. 1, 2

[34] Chuhan Wu, Fangzhao Wu, Lingjuan Lyu, Yongfeng Huang, and Xing Xie. Communication-efficient federated learning via knowledge distillation. *Nature communications*, 13(1):1–8, 2022. 1, 3

[35] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017. 5

[36] Yuhao Zhou, Qing Ye, and Jiancheng Lv. Communication-efficient federated learning with compensated overlap-fedavg. *IEEE Transactions on Parallel and Distributed Systems*, 33(1):192–205, 2021. 5