Practical Hybrid Gradient Compression for Federated Learning Systems

Sixu Hu , Linshan Jiang , Bingsheng He
National University of Singapore
sixuhu@comp.nus.edu.sg, linshan@nus.edu.sg, hebs@comp.nus.edu.sg

Abstract

The high communication cost is a major challenge in the federated learning (FL) training process. Several methods have been proposed to reduce communication costs on the uplink channel, primarily sparsification-based methods, which have overlooked the impact of downlink channels. However, model accuracy and communication cost issues arise when applying them in practical FL applications, especially when the bandwidth is limited both on the uplink and downlink channels. In this paper, we propose a novel secure-FL-compatible hybrid gradient compression framework (HGC) that handles both uplink and downlink communication. Specifically, HGC identifies and exploits three types of redundancies in the FL training process. With proposed optimization methods based on compression ratio correction and dynamic momentum correction, HGC improves the trade-off between communication cost and model performance. The extensive theoretical and empirical analysis demonstrates the effectiveness of our framework in achieving a high compression ratio for both uplink and downlink communications with negligible loss of model accuracy, surpassing the state-of-the-art compression methods.

1 Introduction

Federated learning (FL) [McMahan *et al.*, 2017] enables collaborative learning of a global model from multiple clients without directly communicating each client's private data, by passing model parameters, gradients, or other information about the models. The high communication cost is one of the key bottlenecks in FL training processes, especially for geographically distributed settings required by federated learning. In practice, their bandwidth is often limited to less than 100Mbps [Yang *et al.*, 2019; Philippenko and Dieuleveut, 2022], or it is simultaneously charged, such as 4G/5G networks. Since the communication of uncompressed FL often take gigabytes or even terabytes through iterations, it can easily cost over 80% of the total training time [Xu *et al.*, 2021; Chen *et al.*, 2020] and waste bandwidth resources. Thus, reducing the communication cost in the FL training process is essential for practical deployment of FL systems.

Methods that utilize sparsification [Aji and Heafield, 2017; Lin *et al.*, 2018; Stich *et al.*, 2018; Chen *et al.*, 2020], quantization [Wen *et al.*, 2017; Alistarh *et al.*, 2017; Bernstein *et al.*, 2018; Wu *et al.*, 2022], and their combinations [Basu *et al.*, 2020; Sattler *et al.*, 2020] are shown to be the most effective methods for reducing the communication costs in FL training processes. However, they only focus on compressing the messages in the uplink channel (i.e., model/gradients sent from clients to the server). Although the downlink channel is generally considered to be less bandwidth-limited when compared to the uplink channel [Yue *et al.*, 2022; Philippenko and Dieuleveut, 2022], their bandwidth difference is still within the same order of magnitude. Even within the same cluster, regular Ethernet switch usually provides 1Gbps bandwidth at maximal [Wen *et al.*, 2017], while mobile devices are more limited, averaging at 33.88/9.75 Mbps downlink/uplink in the U.S. in 2019 [Zhang *et al.*, 2020]. In this paper, we show that under a 1Gbps network, with a model having tens of megabytes, the downlink communication can still dominate the training process by taking more than 50% of the training time, even when the uplink communication is sufficiently compressed, as shown in the experiments in Section 5. Therefore, compression algorithms that apply to both uplink and downlink channels are essential to effectively reduce the communication costs in FL training.

To achieve bi-directional compression, a naive idea is to simply apply the same compression method on the downlink channel that is originally used for uplink compression, as employed by existing works [Tang *et al.*, 2019; Zheng *et al.*, 2019; Sattler *et al.*, 2020]. However, these algorithms do not efficiently utilize gradient information. First, they tend to over-compress the gradients. Since each gradient is compressed twice in a communication round, only partial information gathered from the clients is used for updating the global model, consequently degrading the quality of the global model. Second, they only focus on compressing individual clients' gradients but overlook the potential for further reducing the communication cost by utilizing redundancies between clients and across the temporal dimension. Thus, the quality of their global models is not satisfactory. As an example of these two inefficiencies, STC [Sattler *et al.*, 2020] only achieves 83.82% accuracy in the CIFAR-10 task when the sparsification ratio is 0.1% in our reproduction, while the accuracy for the baseline FedAvg algorithm in the same setting is 90.52%.

In short, existing gradient compression algorithms for FL have deficiencies in balancing communication and model accuracy. It involves two aspects: 1) Optimizing communication cost in both the uplink and downlink directions to prevent loss of model quality or convergence rate due to over-compression in the downlink channel. 2) Reduce repetitive or similar information within the gradient of one client, across different clients, and among consecutive gradients updates, to achieve superior compression ratios. We name these repetition intrinsic, cross-client, and temporal redundancy.

To address these issues, we propose a Hybrid Gradient Compression (HGC) algorithm in this paper that utilizes combined compression techniques, including sparsification, quantization, and encoding, to optimize the trade-off between model quality and communication cost. More specifically, we first apply sparsification on the gradients with a shared sparsification mask. Then, the sparsified gradients are quantized and entropy-coded to maximize the compression ratio. These techniques maximizes the compression ratio in both uplink and downlink channels by exploiting the redundancies we have identified, reducing the total size of gradients and metadata such as sparsification masks transmitted.

Our contributions can be summarized as follows.

- We design a FL gradient compression algorithm that operates efficiently on uplink and downlink channels simultaneously and supports existing secure FL techniques. To the best of our knowledge, we are the first to identify and utilize the three types of redundancies for FL gradient compression to achieve a state-of-the-art compression ratio.

- We propose novel optimization methods, namely compression ratio correction and dynamic momentum correction. They effectively improve our compression algorithm to enable efficient trade-off between communication cost and model performance.

- With theoretical analysis of our algorithm, we provide both the convergence guarantees of a generalized gradient compression system. We also provide guidelines for selecting its components and parameters.

- With empirical analysis of our algorithm, we show that our algorithm can achieve a compression ratio much higher than state-of-the-art methods without loss of model accuracy. We also show that our algorithm performs well in various FL scenarios, including skewed data distribution, large-scale training, and secure FL.

In the following part of this paper, we first formalize the problem and elaborate on the backgrounds of gradient compression in Section 2. In Section 3, we present the details of our compression algorithm. Section 4 and Section 5 present the theoretical and empirical analysis, respectively. Finally, we summarize and conclude our work in Section 6.

2 Backgrounds

The FL problem with N clients collaboratively training on a global model $\mathbf{w}$ can be formalized as, where $\mathbf{w}_i$ are the model weight of client i ($i \in [N]$). The weight update rules of its uplink (from client to server) and downlink (from server to client) channels are should be:

$$\mathbf{w}^t = \mathbf{w}^{(t-1)} - \eta \frac{1}{N} \sum_{i=1}^{N} \nabla f(\mathbf{w}_i^{(t-1)}), \mathbf{w}_i^t = \mathbf{w}^t, \forall i \quad (1)$$

To compress the gradient, we apply compressors to the communication process:

$$\begin{aligned} &\mathbf{w}^t = \mathbf{w}^{(t-1)} - \eta \frac{1}{N} \sum_{i=1}^{N} \text{comp}_{\text{up}} \left(\nabla f(\mathbf{w}_i^{(t-1)}) \right), \\ &\mathbf{w}_i^t = \text{comp}_{\text{down}} \left(\mathbf{w}^t \right), \forall i \end{aligned} \quad (2)$$

The main target of this paper is to find good compressors comp_{up} and $\text{comp}_{\text{down}}$ that balance well between the model quality, communication cost, and computation cost. Note that the non-compressed algorithm can be viewed as a special case of Equation 2, where $\text{comp}_{\text{up}} = \text{comp}_{\text{down}} = f : x \mapsto x$ (the identity function).

Related works can be categorized based the on compression techniques they use. While gradient compression has a broad range of techniques, we specifically focus on those that are particularly relevant to our work and align with our focus on compression in FL settings.

Sparsification. Sparsification algorithms utilize the sparsity of gradients and only transmit part of them using sparsification masks. The top-k mask selection method [Aji and Heafield, 2017; Alistarh *et al.*, 2018] is widely used due to its implementation simplicity and high convergence efficiency. Top-k selects and transmits k elements in the gradient sorted by their absolute values. Other index selection methods such as random-k [Stich *et al.*, 2018] and sketch-based [Ivkin *et al.*, 2019] methods have also been widely studied, but they do not show efficiency superior to top-k-based methods.

A typical example of top-k-based compression algorithms is Deep Gradient Compression (DGC) [Lin *et al.*, 2018]. It is a widely tested and compared sparsification algorithm. It updates each client's top-k mask in every communication round and uses the mask to compress the uplink channel communication. Additionally, it performs error feedback to amortize the error inducted by top-k, where the error is stored locally and added to the momentum in subsequent communication rounds. Another noteworthy application of top-k sparsification is ScaleCom [Chen *et al.*, 2020]. It proposes the cyclic top-k (CLT-k) method that explores and utilizes the similarity between the gradients on each client, which shows scalability advantage compared to other algorithms [Strom, 2015; Chen *et al.*, 2018; Lin *et al.*, 2018] that simply apply of top-k on all the clients.

Quantization. Quantization is another way of compressing the gradient by reducing the number of bits of each element in the transmitted gradients. Existing studies such as signSGD [Bernstein *et al.*, 2018] and 1-bit SGD [Seide *et* *al.*, 2014] show that 1-bit quantization is practical and sometimes sufficient in the model training process. More quantization bits are often allowed to better approach the uncompressed gradients by optimizing the selection of quantization levels [Alistarh *et al.*, 2017; Yu *et al.*, 2019]. Other auxiliary methods such as error-feedback mentioned in the sparsification methods can also be applied in the quantization process [Karimireddy *et al.*, 2019; Wu *et al.*, 2018] to further improve the convergence of the training process.

Encoding. Quantization provides a fixed maximum compression ratio. That is, if we compress all 32-bit floats to 1-bit, the overall compression ratio will be $32\times$. If combined with proper lossless encoding, the overall communication cost can be further reduced. Traditional Huffman-coding-based encoding algorithms such as DEFLATE [Deutsch, 1996] and entropy encoding methods like arithmetic encoding [Witten *et al.*, 1987] have shown their efficiency for quantized gradients [Abdi and Fekri, 2019]. Golomb coding [Golomb, 1966] can also reduce the size of indices after sparsification [Sattler *et al.*, 2020; Wu *et al.*, 2022]. Recently, predictive coding techniques [Yue *et al.*, 2022] inspired by video processing have also been proposed, which utilize historical gradient data to further improve the compression ratio of the encoding process.

Hybrid Algorithms. Hybrid approaches utilizing multiple compression methods, especially sparsification and quantization, have been studied to further improve the compression ratio and also achieve other goals such as improving the communication-computation trade-off. Both Qsparse-local-SGD [Basu *et al.*, 2020] and STC [Sattler *et al.*, 2020] combine top-k sparsification and quantization, while SketchML [Jiang *et al.*, 2018] combines sketching and quantization to improve the compression ratio.

However, as we compare the performance of the above methods in Table 1, we find that none of them achieve an optimal state in terms of utilizing data redundancies to achieve a sufficient compression ratio, while also handling bi-directional compression. Most of them only target reducing the intrinsic redundancies of the gradients on individual clients. Meanwhile, they do not ensure compatibility to secure FL, since the gradient size increases when exchanging data between clients. As of today, these algorithms still remain the most performant ones in their respective settings to the best of our knowledge. Other related works such as GossipFL [Tang *et al.*, 2023] either add a different objective other than maximizing the compression ratio or do not provide comparable performance. This leaves us with the possibility to further optimize the entire compression framework.

3 Hybrid Gradient Compression

We design our framework to exploit intrinsic redundancies within a gradient, cross-client redundancies among different clients, and temporal redundancies between consecutive gradients updates to achieve superior compression ratios. The training loop of our algorithm resembles that of FedAvg [McMahan *et al.*, 2017]. The clients train models and compress gradients, which are then sent to the server.

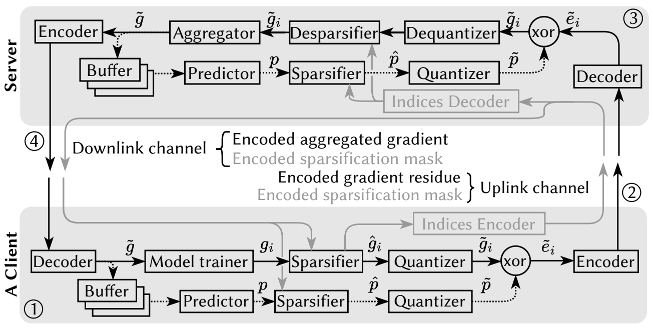


Figure 1: Overview of the HGC Algorithm.

The server decompresses and aggregates the gradients, then broadcasts the gradients to all clients.

Figure 1 illustrates the detailed compression steps on the server and clients, respectively. Note that the communication rounds begin with the model trainer in Figure 1. For each client i, the model trainer computes the gradient g_i. The gradient g_i is then passed through the sparsifier and quantizer, along with the predicted gradient p. The compressed gradient $\tilde{g}_i$ and prediction $\tilde{p}$ are combined using the XOR operator to compute the gradient residue $\tilde{e}_i$ in order to reduce the entropy of the message. The residue $\tilde{e}_i$ is then sent to the server through the uplink channel ② together with the encoded sparsification mask, where both $\tilde{e}_i$ and the mask is encoded with lossless entropy-encoding methods.

The server reverses the client's compression process and mirrors the client's prediction process for restoring the compressed gradient residue $\tilde{e}_i$ to the original compressed gradient $\tilde{g}_i$. After aggregating gradient $\tilde{g}_i$ with the aggregator to obtain $\tilde{g}$, the server uses $\tilde{g}$ to update the global model. Simultaneously, $\tilde{g}$ is sent to both the server's prediction buffer and the clients' prediction buffers to keep all buffers synchronized ④. The local model updates are also performed in this process, completing a communication round.

Compared to existing hybrid gradient compression algorithms, our proposed algorithm has two major differences. First, all clients' sparsifiers and the server's sparsifier share the same user-selectable mask, effectively reducing the indices transmission cost. On the uplink channel, only one client transmits the hard-to-compress mask, while other clients only transmit highly-compressible gradient. On the downlink channel, our approach maintains sparsity of $\tilde{g}$ when aggregating $\tilde{g}_i$ while preventing over-compression in the downlink channel. Second, unlike other studies utilizing gradient residues [Chen *et al.*, 2018; Yue *et al.*, 2022], our algorithm computes the gradient residue $\tilde{e}_i$ after the sparsification and quantization processes. This decouples the prediction process from sparsification and quantization, making it optional and providing us with more control over the lossy compression aspect of the entire algorithm. A more detailed pseudo code implementation is provided in Appendix A.3.

3.1 Hybrid Compression

As shown in Figure 1, our hybrid compression method mainly involves three components: sparsification, quantization, and

Compression algorithm	Methods			Redundancies			Characteristics		
	Sp.	Qt.	Ec.	Intr.	C.C.	T.	Dl.	SFL	$\mathrm{CR}_{\mathrm{up}}$
DGC [Lin *et al.*, 2018]	✓			✓					Medium
ScaleCom [Chen *et al.*, 2020]	✓			✓	✓		✓		Medium
QSGD [Alistarh *et al.*, 2017], signSGD [Bernstein *et al.*, 2018]		✓		✓					Low
Predictive Coding [Yue *et al.*, 2022]		✓	✓	✓		✓			High
Qsparse [Basu *et al.*, 2020], STC [Sattler *et al.*, 2020]	✓	✓	✓	✓			✓[a]		High
HGC (Ours)	✓	✓	✓	✓	✓	✓	✓	✓	High

[*] Abbreviations in the table: Sp.: Sparsification, Qt.: Quantization, Ec.: Encoding, Intr.: Intrinsic redundancy, C.C.: Cross-Client redundancy, T.: Temporal redundancy, Dl.: Downlink compression, SFL: Secure FL compatibility, $\mathrm{CR}_{\mathrm{up}}$: Uplink compression ratio. [a] Only applies to STC.

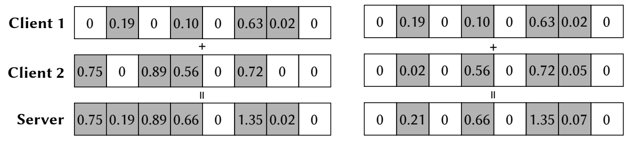
Table 1: Comparison of compression algorithms.

encoding. The gradient of each client g_i is fed through the sparsifier and quantizer, then through the encoder. The sparsifier and quantizer exploit the intrinsic redundancies of the gradient. The mask generation process of the sparsifier utilizes the cross-client redundancy, and then the predictor combined with the encoder utilizes the temporal redundancy. We now elaborate on these three phases separately. The detailed analysis of the redundancies can be found in Appendix A.1.

Sparsification. Both the low efficiency of current compression methods in downlink and secure-aggregation scenarios are caused by sparsification mask mismatch. We illustrate it with a toy example shown in Figure 2a. When performing sparsification independently on different clients, there is a high chance that the sparsification masks differ. Adding these sparse vectors together reduces the result's sparsity, which affects communication efficiency in FL training. However, if the sparsification masks are matched, as shown in Figure 2b, the result shows no sparsity drop. Therefore, well-matched masks can greatly improve the effectiveness of sparsification.

Maintaining a high sparsity level during training is crucial, since the cost of transmitting sparse mask indices is generally higher than the cost of transmitting the sparsified gradient elements. While the sparsified gradient elements can be further compressed through quantization and encoding, the sparse indices themselves are typically incompressible. Therefore, reducing the communication cost of indices is crucial. One effective method to achieve this is by allowing all clients to share indices. By doing so, the communication cost associated with transmitting indices can be significantly reduced.

Apart from making the mask shared, there are no extra restrictions on how the mask should be selected. Either top-k, random-k or even a combination of them can be used. In practice, we find that it is most efficient to make all clients share



(a) Mismatched masks (b) Matched masks

Figure 2: Sparsification with mismatched/matched masks.

one client's top-k mask, minimizing the communication cost of uploading and downloading indices while still maintaining a high global model quality.

Quantization. We choose the quantizer Q with L quantization levels where the l-th level has lower and upper bounds of x_l^{min} and x_l^{max}. We define the index set of l-th level as $\mathcal{I}_l = \{i | x_l^{min} \le x_i < x_l^{max}\}$ and quantize every element in the original tensor x such that:

$$Q(x_i) = \frac{1}{|\mathcal{I}_l|} \sum_{i \in \mathcal{I}_l} x_i \triangleq q_l, \text{ if } x_i \in \mathcal{I}_l, \ \forall i \tag{3}$$

We select the quantization levels by evenly dividing the positive and negative elements in x separately, so that all levels of the same sign have roughly the same number of elements. This quantizer can be viewed as a generalized version of the quantizer used in STC [Sattler *et al.*, 2020], which has been proven effective when combined with the top-k sparsifier.

Encoding. We use the Golomb encoding [Golomb, 1966] to compress the sparsified indices, which can achieve a compression ratio of 2–3× for 32-bit indices [Sattler *et al.*, 2019; Sattler *et al.*, 2020]. We then use arithmetic encoding combined with gzip to compress the quantized and sparsified gradients. Since encoding of the gradient is closely related to the entropy of the data being encoded, we introduce a predictive coding module to reduce such entropy, inspired by a prior predictive encoding work [Yue *et al.*, 2022]. Instead of predicting model weights, the predictor in our approach is expected to generate predictions p that are either positively or negatively correlated with gradient g_i, so that their sparsified and quantized difference $\tilde{e}_i$ can be more biased than $\tilde{g}_i$ to reduce the entropy of the data being encoded. Specifically, in each communication round, we store the aggregated gradient $\tilde{g}_t$ and generate the prediction p_t as shown in Equation 4, similar to the update rule of the Adam optimizer [Kingma and Ba, 2017]. u_t and v_t are initialized as zeros, and β is the running average coefficient.

$$\begin{aligned} u_t &= \beta u_{t-1} + (1-\beta)\tilde{g}_t, \ v_t = \beta v_{t-1} + (1-\beta)\tilde{g}_t^2, \\ p_t &= \frac{u_t}{\sqrt{v_t} + \varepsilon} \end{aligned} \tag{4}$$

The buffers only need to store u_t, v_t and $\tilde{g}_t$, all having the same size of the sparsified gradient. The buffers are synchronized among the server and all clients to make sure that the server and clients get the same predictions. As such, the residue can be correctly restored.

3.2 Optimization Techniques

We consider optimization techniques when implementing HGC, namely sparsification ratio and momentum adjustment, and error feedback. Both of them are essential for improving the model quality in FL when compression is applied.

Dynamic sparsification ratio and momentum adjustment. Through preliminary experiments, we find that sparsification of gradients can lead to the training process to be trapped by saddle point or local minima, since it essentially limits the dimension of optimization direction. We also find that dynamic adjustment of the sparsification ratio helps alleviate this issue and improves the convergence accuracy when used together with error feedback. Further demonstrations of this issue can be found in Appendix A.2. Regarding momentum, multiple studies have addressed its ability to accelerate convergence [Phansalkar and Sastry, 1994; Qian, 1999] and to enable the model to jump out of saddle points [Wang *et al.*, 2020]. Therefore, the adjustment of momentum is also given special consideration in our design.

Although the time and duration for adjusting these two parameters are decided empirically, we find that adjusting these parameters after the optimization process reaches a plateau can minimize the adjustment duration required, and thus incur trivial communication overhead. When using stepped learning rate schedulers, we temporarily lower the compression ratio before and after the learning rate is changed, followed by the momentum change.

Error Feedback. Since the sparsification and quantization used in our algorithm are biased, error feedback is necessary. We adopt the same error feedback algorithm used in existing algorithms [Lin *et al.*, 2018; Karimireddy *et al.*, 2019]. Specifically, we maintain error buffers $\mathbf{m}_i$ that have the same size as the model on each client i and initialize them as $\mathbf{0}$. At the end of each local training, we accumulate the errors locally by updating $\mathbf{m}_i$ as follows: $\mathbf{m}_i \leftarrow \mathbf{m}_i + \hat{\nabla} f_i(\mathbf{w}) - \text{Decompress}(\text{Compress}(\hat{\nabla} f_i(\mathbf{w})))$, where $\hat{\nabla} f_i(\mathbf{w})$ is the gradient on client i. Then, we add them back to the gradients in the next communication round by $\hat{\nabla} f_i(\mathbf{w}) \leftarrow \hat{\nabla} f_i(\mathbf{w}) + \mathbf{m}_i$.

4 Analysis

We theoretically analyze the HGC from two perspectives: the convergence rate and the system utilization (i.e., the communication and computation cost). The precondition for the convergence analysis also provides us with some insights for the selection of the compressor. The detailed proofs of the convergence can be found in Appendix B.

4.1 Selection of the Compressor

Lemma 1 (Contraction property of the compressor)**.** *For any sparsifier S and quantizer Q satisfying $\min(0, 2\mu_l) < Q(\mathbf{x}_i) < \max(0, 2\mu_l)$ for all its quantization level l, compressor $Q \circ S$ satisfies the following contraction property:*

$$\mathbb{E}\|\mathbf{x} - \mathit{comp}(\mathbf{x})\|^2 = \gamma\|\mathbf{x}\|^2 \leq \|\mathbf{x}\|^2 \tag{5}$$

where $0 \leq \gamma \leq 1$ is the contraction coefficient.

Remark. The contraction property of the compressor has been shown to be fundamental to various convergence analyses that involve gradient compression [Alistarh *et al.*, 2018; Stich *et al.*, 2018]. This lemma states that the convergence bound shown in the following Theorem 1 is applicable to a wide range of compressors, quantizers, and their combinations that meet the stated criteria.

4.2 Convergence Analysis

With the contraction property in Lemma 1, the proof in convex optimizations immediately follows the proof of Stich *et al.* [Stich *et al.*, 2018]. The convergence bound in the non-convex case is shown as follows.

Assumption 1 (L-Lipschitz Continuity)**.** *We assume that the target function is L-Lipschitz continuous, i.e., $\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\| \leq L\|\mathbf{x} - \mathbf{y}\|, \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^d$.*

Assumption 2 (Stochastic Gradient Bound)**.** *We assume that the stochastic gradients are unbiased and bounded in every client: $\mathbb{E}\hat{\nabla} f_i(\mathbf{x}) = \nabla f_i(\mathbf{x})$, $\|\hat{\nabla} f_i(\mathbf{x})\| \leq G, \forall i$. It immediately follows that the sum of them is also bounded: $\mathbb{E}\|\frac{1}{n}\sum_{i=1}^{n} \hat{\nabla} f_i(\mathbf{x})\|^2 \leq G^2$.*

Theorem 1 (Convergence bound)**.** *Under Assumption 1 and 2, with a learning rate $\eta_t = \eta_0 = c/\sqrt{T}, \forall t$ where $0 < c < \sqrt{T} \cdot \min\{1, 1/(\sqrt{2}\gamma)\}$ being a constant, the algorithm converges to a local optimum. Specifically, after T communication rounds, we have*

$$\begin{aligned} &\frac{1}{T}\sum_{t=0}^{T} \mathbb{E}\|\nabla f(\mathbf{x}^t)\|^2 \\ &\leq \frac{1}{\sqrt{T}}\left(\frac{4}{c}\left(f(\mathbf{x}^0) - f(\mathbf{x}^*)\right) + \frac{16c\gamma^4 L^2 G^2}{1 - 2\eta_0^2\gamma^2} + 2cLG^2\right) \end{aligned} \tag{6}$$

Indicating that the algorithm implemented by our algorithm has $\mathcal{O}(1/\sqrt{T})$ convergence rate.

Remark. Assumptions 1 and 2 are the assumptions widely used in convergence analysis of machine learning studies [Alistarh *et al.*, 2018; Stich *et al.*, 2018]. Theorem 1 shows that the convergence rate is at least the same as FedAvg (i.e. $\mathcal{O}(1/\sqrt{T})$), which can be further verified in the experiments in Section 5.

4.3 Communication and Computation Cost

In the worst-case scenario, assuming all data after sparsification and quantization are incompressible, the uplink compression ratio of one communication round, with N clients, a top-k sparsifier and quantization bits of q (assuming 32-bit indices and gradients) applied to a model with M parameters, the uplink compression ratio of one communication round will be $32MN/((Nq + 32)k)$ and the downlink compression ratio $M/2k$. However, the actual situation can deviate significantly from this worst-case due to multiple factors influencing the compression ratio, including the effectiveness of the predictor and the entropy encoder. When considering end-to-end compression ratios, additional factors come into play, such as the sparsification ratio warm-up method and the

Task	Algorithm	Max CR*			Total CR*			Speedup	Accuracy / Perplexity
		Up	Down	Total	Up	Down	Total		
ResNet-18 CIFAR-10	FedAvg	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**90.52%**
	DGC	895.57	1.00	2.00	588.83	1.00	2.00	2.11	89.86%
	signSGD	32.07	1.00	1.94	31.96	1.00	1.94	1.80	90.34%
	STC	793.81	1078.11	914.37	687.31	**925.35**	**788.76**	4.78	83.82%
	HGC	**15286.02**	**2911.81**	**4891.79**	**1429.39**	265.78	448.22	**6.27**	90.26%
LSTM PTB	FedAvg	1.00	1.00	1.00	1.00	1.00	1.00	1.00	83.84
	DGC	778.35	1.00	2.00	263.44	1.00	1.99	2.06	84.15
	signSGD	32.05	1.00	1.94	31.92	1.00	1.94	1.71	85.94
	STC	1227.00	1721.66	1432.61	519.85	**811.02**	**632.94**	9.43	118.92
	HGC	**29313.71**	**5262.80**	**8922.38**	**792.49**	388.55	520.84	**11.77**	**80.84**

* "Up", "Down", and "Total" denote uplink, downlink, and total compression ratio respectively.

Table 2: Communication and computation costs.

adjustment of the sparsification ratio. Therefore, it is necessary to measure these ratios empirically.

The additional computational overhead required for compression is negligible compared to the training time, as it only involves a few basic operations such as sorting, addition, and bitwise operations on the gradients. On the contrary, the reduction in gradient size significantly reduces the server's workload, as fewer data from each client needs to be processed. A formalized definition and analysis of the communication and computation costs can be found in Appendix 4.3.

5 Experiments

We use FedAvg [McMahan *et al.*, 2017] as the baseline algorithm and evaluate our method using ResNet-18 [He *et al.*, 2016] on CIFAR-10 [Krizhevsky *et al.*, 2009] and two-layer LSTM [Press and Wolf, 2017] on Penn-TreeBank (PTB) [Marcus *et al.*, 1993]. Unless otherwise specified, we deploy the FL system with four clients with a learning rate of 0.1 for CIFAR-10, and a learning rate of 20 for PTB. We reduce the learning rate by a factor of 0.1 at 50% and 75% of the training process. For algorithms involving sparsification, we use the same sparsification ratio warm-up technique to speed up the training process. Specifically, we exponentially step down the sparsification ratio from 0.25 to the target in 300 iterations. We conduct experiments on a server with two Intel Xeon Gold 6248 CPUs @ 2.50GHz, 192GB of memory @ 3200MHz, and four Nvidia Tesla V100 GPUs.

5.1 Comparison with Other Methods

In addition to the FedAvg baseline, we compare our algorithm with DGC [Lin *et al.*, 2018], EF-signSGD [Karimireddy *et al.*, 2019], and STC [Sattler *et al.*, 2020], which represent the current state-of-the-art algorithms of sparsification, quantization, and hybrid algorithms, respectively. To ensure fair comparisons, we set the sparsification ratio to 0.1% for all algorithms and use 1-bit quantization for all quantizers. We apply the same encoders for indices and data to all algorithms where possible.

In the baseline FedAvg experiments, the amount of data transmitted is identical for both uplink and downlink channels. For the CIFAR-10 task, the size per client per communication round is 44.70MB, resulting in a total data transmitted of 8760.48GB. For the PTB task, the size per client per communication round is 104.50MB, and the total data transmitted is 4915.84GB. We can see that the huge communication volume can be problematic, when the bandwidth is limited on the uplink and downlink channels.

Table 2 presents the communication costs of the tested algorithms, measured by the maximum compression ratio per client per communication round (i.e., Max CR) and the end-to-end compression ratio (i.e., Total CR), separately for the uplink and downlink channels. We also report the speedup achieved by our algorithm compared to FedAvg, assuming a 1000Mbps uplink and downlink bandwidth for the server.

From Table 2 we observe that: 1) The downlink channel is essential for total compression ratio, but over-compression on this channel impairs model quality. 2) Our algorithm achieves the best in maximum compression ratios and total uplink compression ratios without losing accuracy. The total downlink compression ratio is lower than the STC algorithm since it also quantizes the downlink channel while our algorithm doesn't. However, STC does not achieve a convergence accuracy comparable to the baseline due to its over-compression, and the total speedup is not mitigated by the extra downlink compression. Detailed breakdown analysis of accuracy and compression ratio can be found in Appendix C.1.

5.2 Adaptiveness to Federated Learning

To evaluate the compatibility of our algorithm and other FL features, we perform experiments with changes to FL aspects as shown in prior surveys and studies [Li *et al.*, 2022; Yue *et al.*, 2022], namely non-i.i.d (non-identical and independently distributed) datasets, large number of clients, and secure FL.

Figure 3 shows each algorithm's performance under heterogeneously distributed datasets. Although the model quality degrades in this scenario, our algorithm still converges to a higher accuracy compared to other algorithms under non-i.i.d settings, indicating that our algorithm is more robust against heterogeneously distributed data. The HGC + FedProx result also demonstrate our algorithm's compatibility with other op-

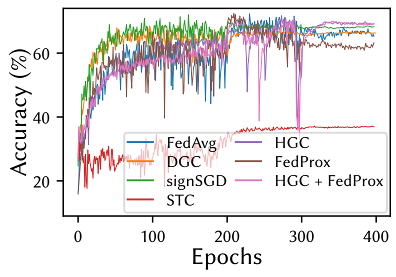


Figure 3: Model accuracy in non-i.i.d settings (ResNet-18 on CIFAR-10).

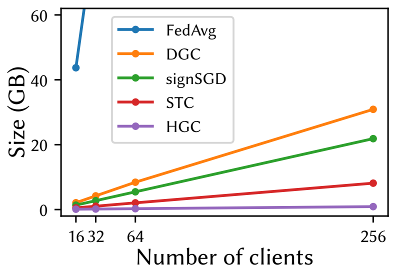


Figure 4: Uplink communication cost w.r.t. The number of clients (CNN on FEMNIST).

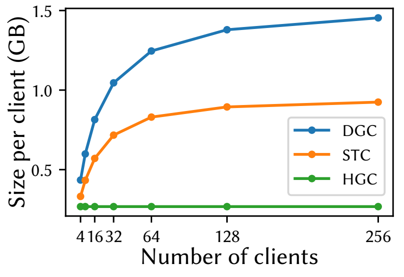


Figure 5: Secure aggregation overhead (CNN on FEMNIST).

timization algorithms like FedProx [Li *et al.*, 2020].

The experiments in Figure 4 and 5 are performed with a 4-layer CNN on the FEMNIST [Caldas *et al.*, 2018] dataset. Since the FEMNIST dataset is collected from 3550 different users, it more closely simulates real-life scenarios. We sample users from the dataset equal to the number of clients and assign each user's data to a client. The result in Figure 4 shows that all algorithms, including the baseline, scale linearly with the number of clients, but our algorithm saves significantly more communication bandwidth with clients.

When secure aggregation is applied to the FL algorithms, the total communication cost of other algorithms scales super-linearly with the number of clients due to additional data being exchanged between clients. As shown in Figure 5, since our algorithm utilizes cross-client redundancy that applies the same mask to all clients, the cost per client remains constant with the number of clients changed as opposed to other sparsification-based algorithms communication. All the above results show that our compression algorithm can be more efficiently adapted to various FL scenarios compared to other compression algorithms. Detailed setup and analysis of these experiments can be found in Appendix C.2.

5.3 Ablation Studies

To explore the effects of various hyperparameters and gain deeper insights into the internals of our algorithm, we conduct a detailed analysis of the maximum uplink gradient compression ratio. This ratio is categorized into three parts, which makes use of the three types of redundancies discussed in the backgrounds. These parts are denoted as CR_i, CR_c, CR_t. By manipulating the algorithm setup, we evaluate these ratios alongside other relevant metrics. A subset of the results is presented in Table 3 and 4.

$(k/M,\, q)^*$	$\text{CR}_i^{\max}$	$\text{CR}_c^{\max}$	$\text{CR}_t^{\max}$	$\text{CR}_{\text{up}}^{\max}$
0.001, 1	1898.46	3.40	2.37	15286.02
0.01, 1	197.77	3.37	4.47	2982.73
0.001, 2	1641.04	3.06	1.30	6506.96

* $(k/M, q)$ stands for (sparsification ratio, quantization bits). $\text{CR}_i^{\max}$, $\text{CR}_c^{\max}$, $\text{CR}_t^{\max}$ are the intrinsic, cross-client and temporal component of $\text{CR}_{\text{up}}^{\max}$.

Table 3: Compression ratio decomposition.

$(k/M,\, q)$	$\overline{\text{CR}_{\text{up}}}^*$	Speedup	Accuracy
0.001, 1	1429.39	6.27	90.26%
0.01, 1	401.00	2.08	90.84%
0.001, 2	1239.48	4.27	89.36%

* $\overline{\text{CR}_{\text{up}}}$ denotes overall uplink compression ratio of the entire training process.

Table 4: Compression ratio decomposition.

From Table 3 we observe that in our algorithm, the utilization of the intrinsic redundancy contributes most to the total compression ratio, while the other two types of redundancies help amplify it. Table 4 shows that both quantization and sparsification have a significant impact on the overall compression ratio and speedup, but they affect the system in different ways. Furthermore, we notice that the optimal hyper-parameters settings for balancing the communication and model accuracy are different under different scenarios. Appendix C.3 shows more detailed evaluations of the impact and effectiveness of each component on resource utilization and model accuracy.

6 Conclusion

In this paper, we present a novel hybrid gradient compression algorithm for federated learning, which effectively compresses the messages both on the uplink and downlink in the FL training process by leveraging three types of redundancies. Through theoretical analysis, we show that our approach achieves convergence at least as fast as the conventional FedAvg method. Furthermore, our empirical studies show the high performance of our algorithm in complex scenarios, including skewed data distribution, large-scale training, and secure FL. Our algorithm achieves a high compression ratio and offers valuable insights for future implementations of compression algorithms in FL.

Acknowledgments

This research is supported by the National Research Foundation, Singapore and DSO National Laboratories under the AI Singapore Programme (AISG Award No: AISG2-RP-2020-018), and the National Research Foundation, Singapore and Infocomm Media Development Authority under its Trust

Tech Funding Initiative. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore and Infocomm Media Development Authority.

References

[Abdi and Fekri, 2019] Afshin Abdi and Faramarz Fekri. Nested dithered quantization for communication reduction in distributed training. *arXiv:1904.01197*, 2019.

[Aji and Heafield, 2017] Alham Fikri Aji and Kenneth Heafield. Sparse Communication for Distributed Gradient Descent. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 440–445. Association for Computational Linguistics, 2017.

[Alistarh *et al.*, 2017] Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. QSGD: Communication-efficient SGD via gradient quantization and encoding. In *Advances in Neural Information Processing Systems 30*, pages 1709–1720. Curran Associates, Inc., 2017.

[Alistarh *et al.*, 2018] Dan Alistarh, Torsten Hoefler, Mikael Johansson, Sarit Khirirat, Nikola Konstantinov, and Cédric Renggli. The convergence of sparsified gradient methods. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, pages 5977–5987. Curran Associates Inc., 2018.

[Basu *et al.*, 2020] Debraj Basu, Deepesh Data, Can Karakus, and Suhas N. Diggavi. Qsparse-local-SGD: Distributed SGD with quantization, sparsification, and local computations. *IEEE Journal on Selected Areas in Information Theory*, 1(1):217–226, 2020.

[Bernstein *et al.*, 2018] Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Animashree Anandkumar. signSGD: Compressed Optimisation for Non-Convex Problems. In *Proceedings of the 35th International Conference on Machine Learning*, pages 560–569. PMLR, 2018.

[Caldas *et al.*, 2018] Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečnỳ, H Brendan McMahan, Virginia Smith, and Ameet Talwalkar. Leaf: A benchmark for federated settings. *arXiv:1812.01097*, 2018.

[Chen *et al.*, 2018] Chia-Yu Chen, Jungwook Choi, Daniel Brand, Ankur Agrawal, Wei Zhang, and Kailash Gopalakrishnan. AdaComp : Adaptive residual gradient compression for data-parallel distributed training. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), 2018.

[Chen *et al.*, 2020] Chia-Yu Chen, Jiamin Ni, Songtao Lu, Xiaodong Cui, Pin-Yu Chen, Xiao Sun, Naigang Wang, Swagath Venkataramani, Vijayalakshmi (Viji) Srinivasan, Wei Zhang, and Kailash Gopalakrishnan. ScaleCom: Scalable sparsified gradient compression for communication-efficient distributed training. volume 33, pages 13551–13563. Curran Associates, Inc., 2020.

[Deutsch, 1996] L. Peter Deutsch. DEFLATE Compressed Data Format Specification. Request for Comments RFC 1951, Internet Engineering Task Force, 1996.

[Golomb, 1966] Solomon Golomb. Run-Length Encodings (Corresp.). *IEEE Transactions on Information Theory*, 12(3):399–401, 1966.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778. IEEE, 2016.

[Ivkin *et al.*, 2019] Nikita Ivkin, Daniel Rothchild, Enayat Ullah, Vladimir Braverman, Ion Stoica, and Raman Arora. Communication-efficient distributed SGD with sketching. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, number 1178, pages 13142–13152. Curran Associates Inc., 2019.

[Jiang *et al.*, 2018] Jiawei Jiang, Fangcheng Fu, Tong Yang, and Bin Cui. SketchML: Accelerating Distributed Machine Learning with Data Sketches. In *Proceedings of the 2018 International Conference on Management of Data*, SIGMOD '18, pages 1269–1284. Association for Computing Machinery, 2018.

[Karimireddy *et al.*, 2019] Sai Praneeth Karimireddy, Quentin Rebjock, Sebastian Stich, and Martin Jaggi. Error Feedback Fixes SignSGD and other Gradient Compression Schemes. In *Proceedings of the 36th International Conference on Machine Learning*, pages 3252–3261. PMLR, 2019.

[Kingma and Ba, 2017] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2017.

[Krizhevsky *et al.*, 2009] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, 2009.

[Li *et al.*, 2020] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. In *Proceedings of Machine Learning and Systems*, volume 2, pages 429–450, 2020.

[Li *et al.*, 2022] Qinbin Li, Yiqun Diao, Quan Chen, and Bingsheng He. Federated learning on non-IID data silos: An experimental study. In *IEEE International Conference on Data Engineering*, 2022.

[Lin *et al.*, 2018] Yujun Lin, Song Han, Huizi Mao, Yu Wang, and Bill Dally. Deep Gradient Compression: Reducing the Communication Bandwidth for Distributed Training. In *International Conference on Learning Representations*, 2018.

[Marcus *et al.*, 1993] Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993.

[McMahan *et al.*, 2017] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 1273–1282. PMLR, 2017.

[Phansalkar and Sastry, 1994] Vijay V Phansalkar and P Shanti Sastry. Analysis of the back-propagation algorithm with momentum. *IEEE Transactions on Neural Networks*, 5(3):505–506, 1994.

[Philippenko and Dieuleveut, 2022] Constantin Philippenko and Aymeric Dieuleveut. Bidirectional compression in heterogeneous settings for distributed or federated learning with partial participation: Tight convergence guarantees. *arXiv:2006.14591*, 2022.

[Press and Wolf, 2017] Ofir Press and Lior Wolf. Using the output embedding to improve language models. *arXiv:1608.05859*, 2017.

[Qian, 1999] Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural Networks*, 12(1):145–151, 1999.

[Sattler *et al.*, 2019] Felix Sattler, Simon Wiedemann, Klaus-Robert Muller, and Wojciech Samek. Sparse Binary Compression: Towards Distributed Deep Learning with minimal Communication. In *2019 International Joint Conference on Neural Networks*, pages 1–8. IEEE, 2019.

[Sattler *et al.*, 2020] Felix Sattler, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek. Robust and Communication-Efficient Federated Learning From Non-i.i.d. Data. *IEEE Transactions on Neural Networks and Learning Systems*, 31(9):14, 2020.

[Seide *et al.*, 2014] 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 *Interspeech 2014*, pages 1058–1062. ISCA, 2014.

[Stich *et al.*, 2018] Sebastian U Stich, Jean-Baptiste Cordonnier, and Martin Jaggi. Sparsified SGD with memory. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[Strom, 2015] Nikko Strom. Scalable distributed DNN training using commodity GPU cloud computing. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[Tang *et al.*, 2019] Hanlin Tang, Chen Yu, Xiangru Lian, Tong Zhang, and Ji Liu. DoubleSqueeze: Parallel Stochastic Gradient Descent with Double-pass Error-Compensated Compression. In *Proceedings of the 36th International Conference on Machine Learning*, pages 6155–6165. PMLR, 2019.

[Tang *et al.*, 2023] Zhenheng Tang, Shaohuai Shi, Bo Li, and Xiaowen Chu. GossipFL: A decentralized federated learning framework with sparsified and adaptive communication. *IEEE Transactions on Parallel and Distributed Systems*, 34(3):909–922, 2023.

[Wang *et al.*, 2020] Jun-Kun Wang, Chi-Heng Lin, and Jacob Abernethy. Escaping Saddle Points Faster with Stochastic Momentum. In *International Conference on Learning Representations*, 2020.

[Wen *et al.*, 2017] Wei Wen, Cong Xu, Feng Yan, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. TernGrad: Ternary gradients to reduce communication in distributed deep learning. In *Advances in Neural Information Processing Systems 30*, pages 1509–1519. Curran Associates, Inc., 2017.

[Witten *et al.*, 1987] Ian H. Witten, Radford M. Neal, and John G. Cleary. Arithmetic coding for data compression. *Commun. ACM*, 30(6):520–540, 1987.

[Wu *et al.*, 2018] Jiaxiang Wu, Weidong Huang, Junzhou Huang, and Tong Zhang. Error Compensated Quantized SGD and its Applications to Large-scale Distributed Optimization. In *Proceedings of the 35th International Conference on Machine Learning*, pages 5325–5333. PMLR, 2018.

[Wu *et al.*, 2022] Donglei Wu, Xiangyu Zou, Shuyu Zhang, Haoyu Jin, Wen Xia, and Binxing Fang. SmartIdx: Reducing communication cost in federated learning by exploiting the CNNs structures. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(4):4254–4262, 2022.

[Xu *et al.*, 2021] Hang Xu, Kelly Kostopoulou, Aritra Dutta, Xin Li, Alexandros Ntoulas, and Panos Kalnis. DeepReduce: A Sparse-tensor Communication Framework for Federated Deep Learning. In *Advances in Neural Information Processing Systems*, volume 34, pages 21150–21163. Curran Associates, Inc., 2021.

[Yang *et al.*, 2019] Qiang Yang, Yang Liu, Yong Cheng, Yan Kang, Tianjian Chen, and Han Yu. BatchCrypt: Efficient Homomorphic Encryption for Cross-Silo Federated Learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 13(3):1–207, 2019.

[Yu *et al.*, 2019] Yue Yu, Jiaxiang Wu, and Junzhou Huang. Exploring Fast and Communication-Efficient Algorithms in Large-Scale Distributed Networks. In *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, pages 674–683. PMLR, 2019.

[Yue *et al.*, 2022] Kai Yue, Richeng Jin, Chau-Wai Wong, and Huaiyu Dai. Communication-efficient federated learning via predictive coding. *IEEE Journal of Selected Topics in Signal Processing*, 16(3):369–380, 2022.

[Zhang *et al.*, 2020] Xiongtao Zhang, Xiaomin Zhu, Ji Wang, Hui Yan, Huangke Chen, and Weidong Bao. Federated Learning with Adaptive Communication Compression Under Dynamic Bandwidth and Unreliable Networks. *Information Sciences*, 540:242–262, 2020.

[Zheng *et al.*, 2019] Shuai Zheng, Ziyue Huang, and James Kwok. Communication-Efficient Distributed Blockwise Momentum SGD with Error-Feedback. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.