



WP9 Scalable federated learning using pre-trained models

D9.3 Report on cost reductions for new FL algorithms

<https://www.fluteproject.eu/>



Funded by
the European Union

This project has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No 101095382. Views and opinions expressed are, however, those of the author(s) only and do not necessarily reflect those of the European Union or the HaDEA. Neither the European Union nor the granting authority can be held responsible for them.

Grant Agreement No.: 101095382
Deliverable: D9.3 Report on cost reductions for new FL algorithms

Project Start Date: 01/05/2023
Coordinator: INRIA

Duration: 36 months

Deliverable No:	D9.3
WP No:	9
WP Leader:	Siemens
Due date:	30/10/2025
Delivery date:	30/10/2025

Dissemination Level:

PU	Public Use	X
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

DOCUMENT SUMMARY INFORMATION

Project title:	Federated Learning and mUlti-party computation Techniques for prostatE cancer
Short project name:	FLUTE
Project No:	101095382
Call Identifier:	HORIZON-HLTH-2022-IND-13
Thematic Priority:	HORIZON-HLTH-2022-IND-13
Type of Action:	HORIZON Research and Innovation Actions
Start date of the project:	01/05/2023
Duration of the project:	36 months
Project website:	https://www.fluteproject.eu/

D9.3 Report on cost reductions for new FL algorithms

Work Package:	WP9 Scalable federated learning using pre-trained models
Deliverable number:	D9.3
Deliverable title:	Report on cost reductions for new FL algorithms
Due date:	30/10/2025
Actual submission date:	30/10/2025
Authors:	Alexandru Serban
Dissemination Level:	PU
No. pages:	61
Authorized (date):	10/10/2025
Responsible person:	Jan Ramon
Status:	Final

Revision history:

Version	Date	Author	Comment
v.0.1	10/10/2025	Alexandru Serban	First draft including the classification and description of the methods
v.1.0	25/10/2025	Carlota Cañamero Herrero, Jan Ramon	Internally reviewed version
v.1.0	27/10/2025	Montse Pardas	Internally reviewed version
v.2.0	30/10/2025	Alexandru Serban	Final version after suggested changes

Quality Control:

	Who	Date
Checked by internal reviewer	Carlota Cañamero Herrero, GRAD	15/10/2025
Checked by internal reviewer	Montse Pardas, GRAD	27/10/2025
Checked by Project Coordinator	Jan Ramon, INRIA	30/10/2025

COPYRIGHT

©Copyright by the FLUTE consortium, 2023-2026.

This document contains material, which is the copyright of FLUTE consortium members and the European Commission, and may not be reproduced or copied without permission, except as mandated by the European Commission Grant Agreement no. 101095382 for reviewing and dissemination purposes.

ACKNOWLEDGEMENTS

FLUTE is a project that has received funding from the European Union's Horizon Europe research and innovation programme under Grant Agreement No 101095382. Please see project URL <https://www.fluteproject.eu/> for more information.

The partners in the project are . The content of this document is the result of the worked developed by the partners in the context of the project.

DISCLAIMER

The content of the publication herein is the sole responsibility of the publishers and it does not necessarily represent the views expressed by the European Commission or its services. The information contained in this document is provided by the copyright holders "as is" and any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. In no event shall the members of the FLUTE collaboration, including the copyright holders, or the European Commission be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of the information contained in this document, even if advised of the possibility of such damage.

Contents

I	Efficient Federated Fine-tuning for Prostate Cancer Detection	8
1	Introduction	8
2	Background	9
2.1	Foundational Models and Federated Learning	9
2.2	Optimizing Communication Costs of Federated Fine-tuning	11
3	Results from D9.2	12
3.1	Dataset	13
3.1.1	Preprocessing and Normalization	14
3.2	Models	16
3.3	Results from D9.2	17
4	Methods	18
4.1	Top-k Sparsification	19
4.2	Update Frequency Reduction	19
4.3	Delta Encoding	20
4.4	Client Sampling	20
4.5	Hybrid Methods	21
5	Results	21
5.1	Top-k Sparsification	21
5.2	Update Frequency Reduction	22
5.3	Delta Encoding	23
5.4	Client Sampling	25
5.5	Hybrid Methods	26
6	Discussion and Conclusions	27
II	Communication-Efficient Federated learning: New algorithms for Gradient Projection onto Historical Descent Directions	32
1	Introduction	32
2	Setting and State of the Art	33
2.1	Compressors	34
2.2	Algorithm Design	35
3	Two New Algorithms	35
3.1	Our Main Algorithm: ProjFL	36
3.2	Integration of Error Feedback Mechanism	36
4	Theoretical convergence results	37

5	Numerical experiments	39
6	Limitations	42
7	Conclusion and Discussion	43
A	Experimental details and Additional experiments	43
A.1	Experimental details	43
A.2	Overfitting on the Training Set for CIFAR-10	46
A.3	Experiments with $M = 10$ Clients	46
A.4	Additional Experiments with Uplink and Downlink Communication Costs	47
A.5	Accuracy	49
A.6	Evaluation of EF21 and DIANA under different hyperparameters	49
A.7	Variability Across Runs	49
B	Proof of Theorem 4.3	49
C	Proof of Theorem 4.4	55

Summary

This report details the development and implementation of efficient federated fine-tuning algorithms for foundational models, specifically for detecting prostate cancer using image-based data. It also introduces a new novel FL algorithm that leverages gradient projections onto subspaces spanned by historical descent directions – known to both clients and the server – to enable efficient information communication in federated learning with minimal overhead.

Therefore, this report is divided in two parts.

In the first part of the report, following recommendations from D9.1 and D9.2, we investigate multiple methods to decrease the communication costs for additive model fine-tuning, including top-k sparsification, update frequency reduction, delta encoding, and others.

Experiments conducted on a custom dataset from Siemens, using task-specific foundational models, yield key insights for federated learning for prostate cancer detection. Top-k sparsification emerges as the most effective communication optimization strategy, achieving up to 70% parameter reduction with minimal performance degradation by preserving temporal consistency in federated learning. Update frequency reduction introduces significant performance penalties, particularly affecting parameter-efficient methods like LoRA, while delta encoding provides adaptive advantages with natural communication overhead reduction as training progresses. Critical findings include the absence of universally optimal hyper-parameters across optimization techniques, requiring method-specific tuning, and the irreplaceable nature of comprehensive client participation, as even 50% participation rates result in 10% performance loss. Hybrid approaches combining top-k sparsification and delta encoding demonstrate superior efficiency (85% parameter reduction), though with increased implementation complexity. The robustness of 4-bit quantization across all techniques enables practical deployment on resource-constrained edge devices, while practical challenges include method selection complexity, hyperparameter management scalability, and the need for adaptive optimization strategies in heterogeneous federated environments.

In the second part of the report, we present the ProjFL algorithm – a novel approach that projects gradients onto a shared client-server subspace, achieving significant convergence speed improvements while transmitting only one additional scalar per iteration, with an Error Feedback extension for enhanced performance under biased compression.

Theoretical analysis establishes linear convergence to a noise ball for stochastic gradient descent and $O(1/T)$ convergence rate for gradient descent under standard assumptions.

Empirical validation on neural networks (LeNet-5, ResNet-20) demonstrates superior performance over state-of-the-art baselines in final accuracy and stability, with rapid convergence requiring minimal hyperparameter tuning and achieving up to 8× reduction in communication cost compared to existing methods. These improvements remain to be tested on the prostate use-case, where larger foundation-style backbones, multi-modal inputs, and higher class imbalance prevail.

Part I

Efficient Federated Fine-tuning for Prostate Cancer Detection

1 Introduction

WP9 extends the FLUTE platform by integrating Federated learning (FL) algorithms that leverage unlabeled data through techniques such as pre-training, fine-tuning, and more general use of foundational models (FMs) in FL. This extension aims to improve the platform's scalability, allowing it to manage larger data volumes and model sizes more efficiently. By doing so, WP9 seeks to narrow the cost gap between new privacy-preserving machine learning algorithms and more traditional methods, which often overlook the trade-offs associated with cost and scalability.

To evaluate the algorithms developed in WP9, we build upon the existing work in FLUTE for prostate cancer detection. While WP2 and WP5 combine the MRI data with human-engineered features, WP9 transitions to an approach that relies only on image-based data and aims to leverage unlabeled data to overcome potential limitations and improve performance. This involves using pre-trained FMs on unlabeled data, followed by fine-tuning these models with FLUTE data.

This part of the report builds upon D9.1, which reviewed and classified algorithms for training and using FMs in FL, and upon D9.2, which developed fine-tuning and distillation algorithms of FMs in FL. It presents the results and insights derived from reducing the communication costs of federated fine-tuning algorithms for FMs, specifically applied to the prostate cancer use-case within the FLUTE project, as well the results of an independent study on reducing the communication costs of training models in FL.

In summary, D9.1 recommended that for the collaborative training of FMs using FL, partial model pre-training with additive methods as an optimal choice for future integration with Privacy-Enhancing Technologies (PET), as will be done in FLUTE [1]. These methods offer lower complexity and higher efficiency, making them more adaptable to the increased overhead from technologies like encryption for shared parameters. In contrast, methods for entire model pre-training remain challenging to implement, even without PET.

Furthermore, customizing FMs using FL was recommended as the most compelling method for practical applications, such as the FLUTE case study. Fine-tuning, hybrid fine-tuning, and contraction methods such as distillation, compression or quantization offer significant benefits in terms of efficiency and scalability. These methods also facilitate the integration of PET technologies. From the fine-tuning methods, adaptive methods through low-rank adaptation (LoRA) or final stage tuning were found to be the most suitable, as besides scalability and efficiency, they enable a suite of customization options such as client-based resource balancing or personalization [2, 3].

Given these initial conclusions, D9.2 developed and applied these algorithms on the prostate use-case and presented the results, and indicated that federated fine-tuning through adaptive methods using LoRA or adapters in the final stage are the best performing and most efficient in terms of communication costs. In this report, we benchmark several techniques for reducing the communication overhead are on top of adaptive fine-tuning models developed in D9.2. We presents these experiences together with trade-offs associated with their implementation. All experiments were performed on a custom dataset available at Siemens, retrieved from 17 internal collections distributed across 3 geographical regions and consisting of approximately 1393 patients for which a biopsy was performed.

The use-case involves classifying prostate cancer patients using only image-based data. Following D9.2, we explore several paths to make the communication more efficient on the prostate detection use-case, namely top-k sparsification, update frequency reduction, delta encoding, client sampling, as well as hybrid methods.

We find that that top-k sparsification achieves the most effective communication optimization (70% parameter reduction with minimal performance loss), while update frequency reduction significantly penalizes parameter-efficient methods like LoRA. Key findings demonstrate that no universally optimal hyperparameters exist across techniques, comprehensive client participation remains irreplaceable (50% participation causes 10% performance loss), and hybrid approaches can achieve 85% parameter reduction at the cost of increased complexity.

The remainder of this part of the report is organised as follows. We first discuss background information for the methods used in this report, including FMs and FL and communication efficient methods (Section 2) followed by a summary of the main findings from D9.2 (Section 3), an overview of the methodology (Section 4) and the main empirical results (Section 5) We end with a discussion about the findings in the context of FLUTE and conclusions (Section 6).

2 Background

We present a concise introduction to FMs and FL, along with the core techniques employed in D9.2 that form the foundation of this report (Section 2.1). Additionally, we provide background information on the optimization techniques used in this report to further reduce communication costs in federated fine-tuning scenarios (Section 2.2). Comprehensive methodological details and experimental settings are further discussed in Section 4.

2.1 Foundational Models and Federated Learning

FMs are pre-trained on unlabeled data by creating pretext tasks, thereby eliminating the need for annotated data and significantly expanding the datasets available for learning. These models have demonstrated improved performance and robustness across a wide range of image-based machine learning (ML) tasks, reducing the reliance on large, labeled datasets for each specific task [4, 5, 6]. This reduction lowers the costs and efforts associated with data annotation and model training. As FMs continue to demonstrate superior performance and versatility, they are increasingly becoming standard in ML engineering, where starting from an already available pre-trained model is a common practice.

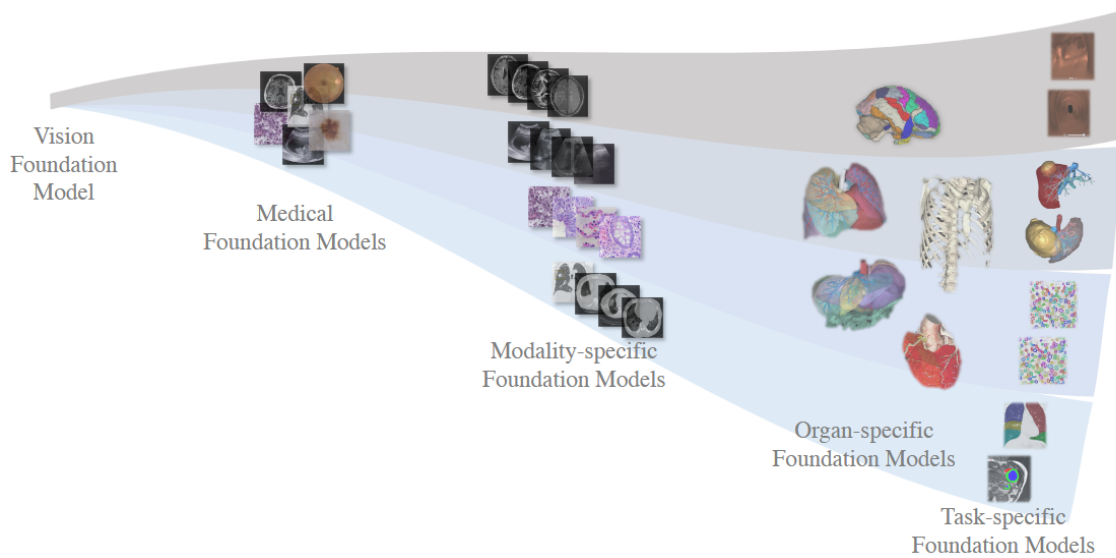


Figure 1: The spectrum of foundational models for medical images, as classified by [7].

In medical image analysis, the spectrum of FM can be categorized based on the data used for pre-training [7]. This ranges from more general medical FMs, which are trained on multiple modalities and organs, to more specific foundational models. The latter are trained on modality-specific data (e.g., MRIs with multiple organs), organ-specific data (e.g., prostate MRIs), and even task-specific data (e.g., lesion-based Federated Models). An illustration of this classification is provided in Figure 1.

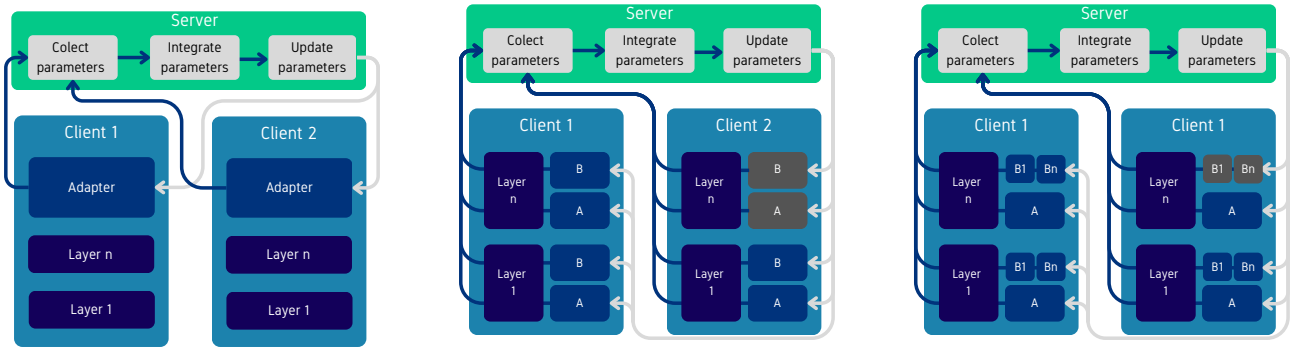
Given that unlabeled data is available for pre-training, more specific FMs such as organ or task-specific models are expected to perform better for a particular task [7], as is the case in FLUTE. This is because these models are tailored to the specific characteristics and requirements of the data that will be also used in the downstream task.

Furthermore, these models are typically smaller in terms of the number of parameters, as they have to learn less data variations and nuances compared to more general models. This reduction in complexity is beneficial for FL settings, where communication costs must be optimized.

To customize FMs for the FLUTE use-case, additive fine-tuning algorithms demonstrated superior performance compared to alternative approaches, as presented in D9.2, establishing them as the optimal strategy for our federated learning setting, and serving as the foundation for the communication optimization experiments presented in this report.

An overview of additive fine-tuning methods is illustrated in Figure 2. Specifically, we explore additive fine-tuning with adapters in the final stage (Figure 2a) and additive fine-tuning using layer-based LoRA adapters (Figure 2b). These methods are further enhanced with quantization techniques to create hybrid approaches.

For the first method, at the start of training, the FM and the newly initialized last layer adapter is distributed to all clients. The FM remains frozen throughout the training process, and only the final layer is involved in FL. During each training epoch, each client performs a forward pass through its data and shares the parameters from the final layer with the server. The server



(a) Additive fine-tuning with adapters in the final stage.

(b) Additive fine-tuning using LoRA adapters.

(c) Additive fine-tuning using asymmetric LoRA.

Figure 2: Overview of additive fine-tuning methods, where the grey boxes mean that the added parameters can be heterogeneous between clients.

aggregates these parameters and sends them back to the clients, repeating this process until convergence.

In the second method, at the beginning of training, the FM and a set of low-rank adapters, chosen for each layer of the FM, are distributed to all clients. During each training epoch, each client performs a forward pass through its data and shares the parameters of the LoRA adapters with the server, which then aggregates the results. While this method increases the number of parameters, it allows for more adaptive and nuanced fine-tuning of the model.

The low-rank adapters are defined by two trainable matrices $Wx = BAx$, where A encodes the input to a lower dimensional rank and B recovers the output dimension of the original W_0 . During the forward pass, each model layer processes the input through the frozen original weights W_0x and the low-rank adapters. The outputs from these paths are summed and passed to the next layer as $W_0x + \Delta Wx$. The Δ parameter serves as a constant scale coefficient.

In the context of transformer-based architectures, which are used in the experiments presented in the report, LoRA is applied specifically to the attention weight matrices, while the MLP modules remain frozen.

2.2 Optimizing Communication Costs of Federated Fine-tuning

Given the outcomes of D9.2 (discussed in more details in Section 3), we employ several methods to further optimize the communication costs in our FL network.

First, we begin by employing top-k sparsification [8, 9] to the shared adapter parameters – a technique that selects only the k largest parameters from the update vectors and communicates exclusively these parameters with the server. In this configuration, all remaining parameters are set to zero, creating a sparse update vector where only the indices and values of the k non-zero parameters are transmitted. This approach can significantly reduce the communication payload while preserving the most influential parameter updates.

Second, we employ update frequency reduction [10] – a communication optimization strategy that decreases the number of communication rounds between clients and the central server

by allowing clients to perform multiple local training epochs before transmitting their model updates. Rather than the conventional approach where clients send updates after each local epoch, this technique enables clients to execute several local optimization steps or complete training epochs on their local datasets before participating in the global aggregation process. While this method can reduce communication overhead, it also introduces a trade-off: improved communication efficiency comes at the cost of increased client drift, where local models diverge more substantially from the global model due to extended local training on heterogeneous data distributions. The effectiveness of this strategy depends on data heterogeneity across clients, local learning rates, and the number of local epochs performed.

Third, we employ delta encoding [11, 12] – a compression technique that transmits only the differences (deltas) between consecutive parameter states rather than sending complete model parameters. Clients maintain a reference model state (typically the last global model received from the server) and compute parameter differences between their locally trained model and this reference point, transmitting only these delta values during communication rounds. This method exploits the observation that model updates are often sparse and incremental, with many parameters changing only slightly between rounds, making delta encoding particularly effective for bandwidth reduction.

Fourth, we investigate client sampling – where only a subset of available clients participates in each communication round, rather than requiring universal client participation. The central server selects a fraction of clients at each round using various sampling strategies: uniform random sampling, importance-based sampling that prioritizes clients with larger datasets or superior connectivity, or stratified sampling that ensures representation across different data distributions or geographical regions. While advanced sampling techniques can incorporate factors such as client reliability, data quality, and computational capacity to optimize training efficiency, client sampling introduces variance in the aggregation process and can exacerbate data heterogeneity effects, requiring careful tuning of learning rates and aggregation weights to maintain stable convergence.

Last, we experiment with hybrid approaches that combine the aforementioned methods with quantization techniques, leveraging the principle that orthogonal compression strategies can yield multiplicative gains in communication efficiency. These hybrid methods exploit the complementary nature of distinct compression techniques, such as pairing delta encoding with client sampling to transmit parameter differences from selected participants, or integrating sparsification with quantization to reduce both the number of transmitted parameters and their bit representation. Such combinations can achieve substantial compression ratios while requiring sophisticated error correction mechanisms to maintain model performance.

All experimental details and implementation specifics are provided in Section 4.

3 Results from D9.2

We introduce the key findings from D9.2 that serve as the foundation for this report and provide the necessary context for the communication optimization experiments. This includes comprehensive details about the dataset employed in our studies (Section 3.1), the models (Section 3.2) and the main experimental results that inform our subsequent optimization

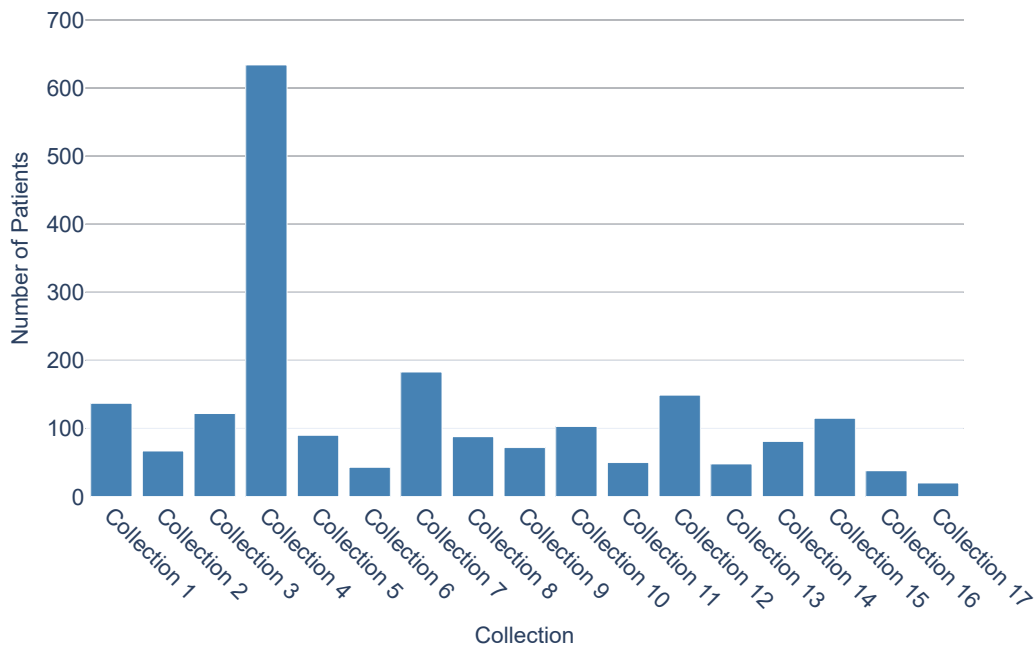


Figure 3: Distribution of patients across vendor collections.

strategies (Section 3.3).

3.1 Dataset

The experiments detailed in this report used an internal dataset sourced from 17 internal collections (representing independent clinics) spread across North America, Europe, and Asia, consisting of T2 MRI acquisitions together with associated ADC and DWI. We illustrate the distribution of patients per collection in Figure 3. The dataset comprised a total of 2,040 patients. Among these, 1,393 patients had a biopsy (Gleason score), and all 2,040 patients had an associated PI-RADS score. As will be discussed in this section, the patients with PI-RADS score are mainly used to balance the collections for non-cancerous patients (i.e., PI-RADS less than three). We note that the datasets across vendors is unbalanced, with one collection (Collection 4) having significantly more patients than the others. This is in line with the distributions expected in FLUTE, and trade-offs stemming from this imbalance will be discussed in this report.

We also provide an illustration of the Gleason scores in Figure 4, which are used to classify patients with benign and malignant tumors. A Gleason score of zero indicates normal tissue architecture (non-cancerous). Score one represents tumors with preserved glandular architecture (well-differentiated), score two shows partially disrupted architecture (moderately differentiated), and score three exhibits severely disrupted architecture with infiltration (poorly differentiated). Clinically significant prostate cancer is considered any score equal or above two, while one is considered to be a malignant nodule that looks similar to normal prostate tissue and is not clinically significant.

We observe that the majority of patients have Gleason scores of one or higher, indicating the presence of a malignant tumor. This imbalanced distribution is expected, as patients who

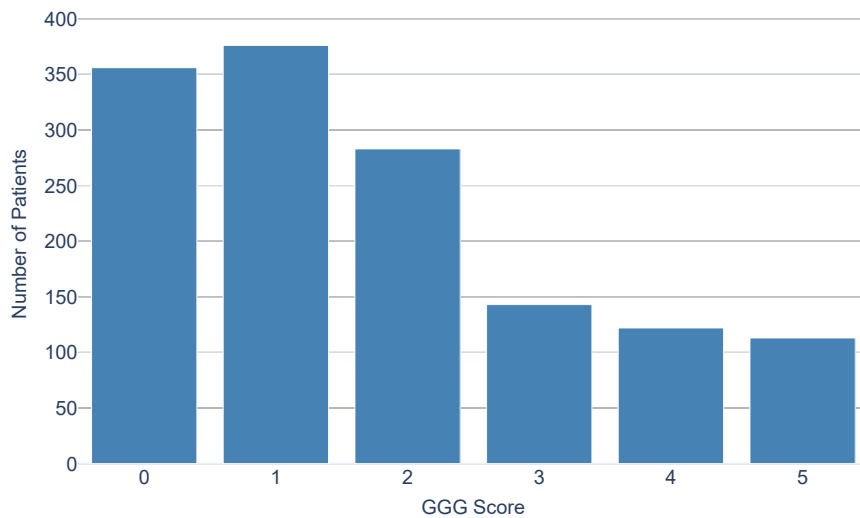


Figure 4: The Gleason grading score distribution in the dataset described in Figure 3.

undergo a biopsy are first screened by a clinician who recommends the procedure based on the suspicion of a malignant tumor. To address this imbalance, as mentioned above and detailed in the following sections, we will include patients where the absence of a malignant tumor is almost certain, given their low PI-RADS score and the recommendation to not undertake a biopsy.

The data was collected using three different scanner manufacturers with varying scanner models. Additionally, due to regional differences and varying acquisition protocols across clinics, we anticipate some variability in the final samples. To examine this, we present the distribution of mean values in the T2 in Figure 5. We notice that certain collections, such as Collection 7 and Collection 15, exhibit distinctly different distributions in terms of both mean and spread. Upon closer examination, we found that these collections are from the same geographical region and use the same scanner vendor. It is probable that these clinics employ a different configuration for their acquisition protocol. Such details will become important when normalizing the data and for experimenting with grouping multiple clinics across FL nodes.

3.1.1 Preprocessing and Normalization

Our goal is to predict the presence of a potentially malignant tumor in a patient using only imaging data, specifically T2, ADC, and DWI, which corresponds to a Gleason score of at least one. For this task we define the positive class as “any histologically confirmed prostate adenocarcinoma” (i.e., Gleason score / Grade Group one or higher), and the negative class as benign/non-cancer tissue. Although clinically significant prostate cancer is typically restricted to cases with Gleason score two or more, we deliberately broaden the positive label to include lower-grade disease. This choice prioritizes sensitivity to any malignant process rather than only those already meeting csPCa criteria, enabling the model to distinguish cancer (regardless of clinical significance) from non-cancer.

To further balance the Gleason score distribution, we also categorize patients with PI-RADS

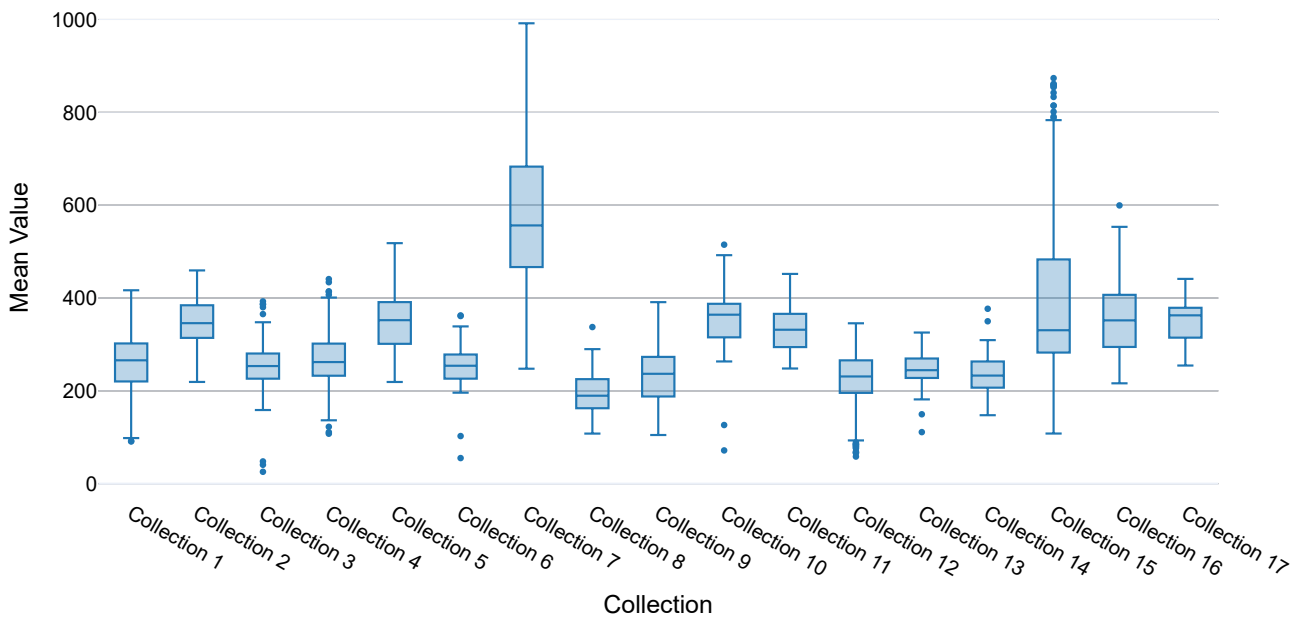


Figure 5: The distribution for the mean values of T2 volumes per collection.

scores of one and two (who do not have a Gleason score) as negative. An illustration of this distribution is provided in Figure 6.

To normalize the T2 volumes, given the distributions illustrated in Figures 5, we use per volume min-max normalization [13], scaling intensities between 0 and 1. Using per volume normalization helps standardize the final distributions across different scanner vendors and clinics.

ADC and DWI require distinct preprocessing because they differ in biophysical meaning and in sources of variability. ADC values are quantitative, expressed in mm^2/s , and their absolute scale carries diagnostic significance (e.g., markedly reduced ADC in highly cellular, aggressive tumors). Applying per-volume min-max normalization would (i) compress clinically relevant low ranges if outliers inflate the max, (ii) introduce sensitivity to noise spikes or susceptibility artifacts, and (iii) destroy cross-patient comparability of threshold-based biomarkers and absolute radiomic descriptors. We therefore adopt a simple division by a fixed constant (as used in prior work [14]) to place values into a numerically stable range while preserving the physical diffusion scale.

In contrast, raw high- b -value DWI signal intensities are not intrinsically standardized: they vary with coil sensitivity, vendor gain settings, TE/TR, gradient performance, and receive profile. Rather than enforcing an arbitrary global min-max, we normalize the high b -value image by the median intensity of the corresponding $b=0$ image, producing a robust relative attenuation measure that is less affected by localized outliers. A further division by a constant coarsely harmonizes distributions across vendors without erasing lesion-specific contrast. This two-step approach reduces domain shift while retaining diffusion-related hyperintensity patterns essential for malignancy characterization.

Overall, our divergent preprocessing strategies reflect a deliberate choice: preserve absolute quantitative information for ADC (supporting established clinical thresholds) and derive a

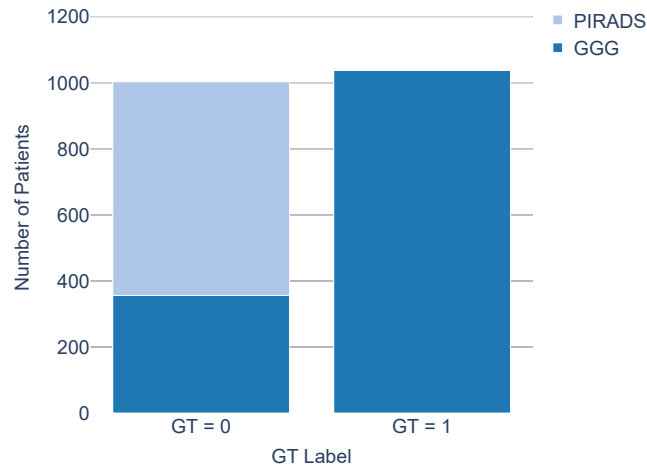


Figure 6: The distribution of ground truth labels based on Gleason score grading (GGG) and PI-RADS.

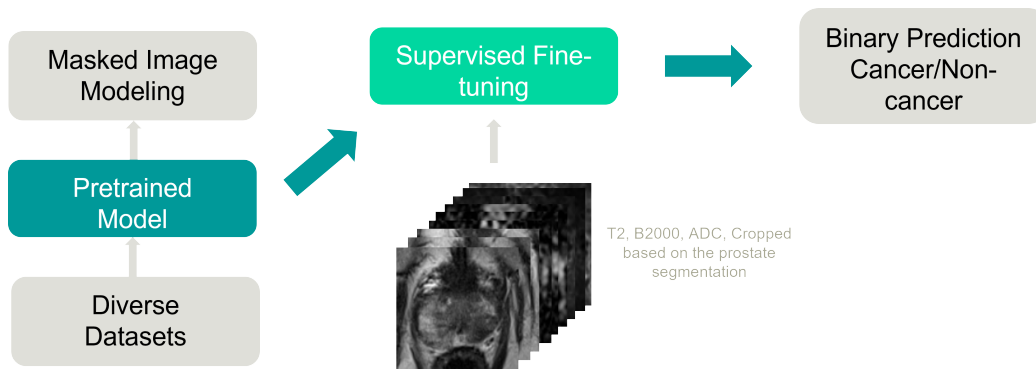


Figure 7: Illustration of training paradigms for pre-training and supervised fine-tuning.

stable, relative contrast representation for DWI to mitigate acquisition variability.

3.2 Models

The models employed were based on SwinViT architectures, initially pre-trained as lesion-specific FMs and subsequently fine-tuned in a federated learning network using the techniques outlined in Section 2.

For pre-training, we employed Masked Image Modelling (MIM), a self-supervised learning technique that reconstructs masked portions of input images. During pre-training, approximately 50% of each input sample was masked, with the model trained to reconstruct the missing information. MIM is recognized as one of the top-performing image-based self-supervised algorithms[15, 16], establishing it as a robust pre-training baseline.

By randomly masking image portions and requiring reconstruction, the model learns contextual relationships between different image regions, developing understanding of visual patterns, textures, object shapes, and spatial positioning that transfers effectively to downstream classification tasks. This approach exploits natural image correlations, enabling strong

Method	Centralized AUROC (%)	FL AUROC (%)
LoRA Rank 16	84.74 (0.025)	86.24 (0.024)
LoRA Rank 32	86.02 (0.024)	82.36 (0.027)
LoRA Rank 64	87.09 (0.024)	83.83 (0.026)
Final-stage	85.22 (0.025)	85.13 (0.025)
Entire model fine-tuning	88.17 (0.023)	83.01 (0.026)

Table 1: Performance and confidence intervals for the lesion-based Swin model fine-tuned in a FL network and centralized. In the centralized setting, all training data is combined into a single collection and node, while in FL, each dataset from Figure 3 is assigned to a separate node in the network. We observe a consistent decrease in performance when running the model in FL, for both fine-tuning with LoRA and with final-stage adapters.

generalization from unlabeled data.

Pre-training employed over 4,000 prostate lesion samples, consistent with current literature where (author?) [17] demonstrated significant performance improvements using approximately 4,400 samples for pre-training. The input data comprised three frames per imaging view (T2, ADC, DWI/B2000), stacked to create nine-channel input images. The first frame is the middle frame in a lesion, taken together with the frames before and after the mid one. Images were cropped around the lesions using existing segmentation masks to focus model attention on the region of interest, then standardized to the same resolution across all vendors through padding and cropping (if the lesion is smaller we apply padding and if it is larger we apply center cropping), as illustrated in Figure 7.

For the fine-tuning stage, the dataset was partitioned using stratified sampling into 80% training, 10% validation, and 10% testing to ensure representative collection distribution. Model performance was assessed using AUROC [18], which effectively balances sensitivity and specificity for binary classification tasks. We employed weighted binary cross-entropy loss [19] with the AdamW optimizer at a learning rate of 10^{-4} [20]. These standard configurations were deliberately selected to ensure reproducibility and enable focus on model performance evaluation without hyperparameter tuning variability.

3.3 Results from D9.2

D9.2 achieved optimal results using federated additive fine-tuning, comparing final-stage optimization against LoRA adapters across all transformer layers. It employed the lesion-based pre-trained model, fine-tuning it with final-stage fine-tuning and various LoRA ranks (16, 32, 64) against centralized baselines. The best results from D9.2 are summarized in Table 1 using AUROC with Newcombe’s Wald Method confidence intervals [21].

Furthermore, D9.2 showed that LoRA rank scaling presented inconsistent federated learning behavior compared to centralized training. While centralized fine-tuning demonstrated consistent improvement from rank 16→32→64 (nearly matching full fine-tuning), federated learning peaked at rank 16 with unpredictable performance at higher ranks. Final-stage additive parameters exhibited greater consistency across centralized and federated settings,

achieving comparable performance to LoRA-16 while requiring significantly fewer parameters, indicating superior efficiency for federated deployment.

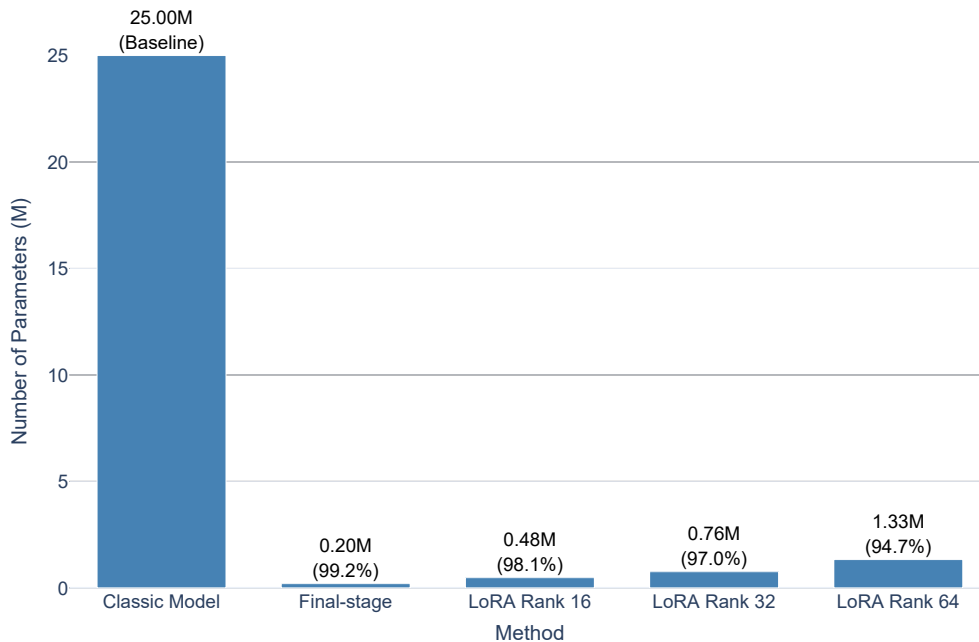


Figure 8: The effect of additive fine-tuning on the model parameters communicated across the network in FL. Fine-tuning with adapters in the final stage proves to be the most efficient, whereas LoRA-64 is the least efficient, as it introduces a larger number of parameters to the model.

Fine-tuning the entire model in FL also leads to a more significant drop in performance. This behavior aligns with existing literature summarized in D9.1, as averaging the entire model across datasets with distinct distributions can result in diminished performance.

Additive federated fine-tuning significantly reduces model parameter sizes and communication overhead, as illustrated in Figure 8. The analysis from D9.2 reveals that LoRA achieves up to 98.1% parameter reduction while final-stage fine-tuning reaches 99.2% reduction. Although final-stage adapters provide maximum parameter efficiency, they don't deliver optimal accuracy improvements. Parameter reduction directly translates to lower federated learning communication costs, with total data transfer ranging from approximately 0.8-7 MB per client per epoch, depending on precision (e.g., float16 vs. float32). This demonstrated the practical advantage of additive fine-tuning for bandwidth-constrained federated deployments.

4 Methods

This section presents the methodological framework and experimental implementation of the communication optimization techniques introduced in Section 2. We provide detailed descriptions of our experimental approaches for top-k sparsification (Section 4.1), update frequency reduction (Section 4.2), delta encoding (Section 4.3), client sampling (Section 4.4), and hybrid method combinations (Section 4.5).

4.1 Top-k Sparsification

Top-k sparsification selects the k largest parameters from the update vectors and communicates exclusively these parameters with the server. All remaining parameters are set to zero, creating a sparse update vector where only the indices and values of the k non-zero parameters are transmitted. For a gradient vector g with dimension d , the top-k sparsification operator $S_k(g)$ can be defined as:

$$S_k(g) = \begin{cases} g_i & \text{if } |g_i| \text{ is among the } k \text{ largest values} \\ 0 & \text{otherwise} \end{cases}$$

We select the k largest parameters by retaining a specified percentage of original parameters, with the optimal percentage varying based on application requirements, performance targets, and acceptable communication reduction levels.

Common sparsification strategies include:

- Aggressive sparsification: 0.1%–1% (maximum communication reduction)
- Moderate sparsification: 1%–10% (balanced trade-off)
- Conservative sparsification: 10%–30% (highest accuracy preservation)

The key influencing factors for selecting k are based on the model characteristics – as larger models typically tolerate aggressive sparsification due to inherent parameter redundancy; the accuracy-communication trade-off – where lower k values reduce communication overhead but may cause convergence instability or accuracy degradation; optimization compatibility – where some optimizers demonstrate greater sparsification robustness (e.g., SGD often outperforms adaptive methods like Adam under sparse conditions) and infrastructure constraints – where network bandwidth and latency limitations determine practical communication budgets, directly influencing feasible sparsification levels for real-world deployments.

Since the choice is application dependent, we experiment with k values, ranging from 1% to 30%. Furthermore, to maintain convergence guarantees, we also implement error feedback mechanisms where the dropped parameters are accumulated locally and added to future updates.

4.2 Update Frequency Reduction

Update frequency reduction [10] reduces communication rounds by enabling clients to perform multiple local training epochs before transmitting updates, effectively accumulating gradients locally and synchronizing globally every n epochs.

The optimal epoch count n depends on several factors: data heterogeneity (non-IID data increases client drift risk with excessive local steps), computational capacity (powerful clients support more local computation), network constraints (poor bandwidth incentivizes reduced frequency), model complexity (complex models require frequent synchronization), and performance requirements (excessive reduction slows convergence).

In this report we determine optimal update frequency through experimentation across $n = 2 - 10$, monitoring: (i) gradient staleness from outdated global model versions, and (ii) model drift by comparing local versus global performance metrics.

4.3 Delta Encoding

Delta encoding [11, 12] transmits only the differences (deltas) between consecutive parameter states rather than sending complete model parameters. At each training round the central server sends the current global model W_t to the clients. The clients perform local training on the local datasets for a number of steps, resulting in an updated local model $W_{t,\text{local}}$. Instead of sending $W_{t,\text{local}}$ directly, each client calculates the difference between its locally trained model and the global model it started with: $\Delta W_{t,\text{client}} = W_{t,\text{local}} - W_t$. This $\Delta W_{t,\text{client}}$ is the model update (model delta) that the client intends to contribute. This vector is often much sparser.

The effectiveness is measured by the sparsity of the delta vector (i.e., how many elements in the delta vector are zero or near-zero) and the magnitude of the non-zero delta values (i.e., how small are the actual changes).

4.4 Client Sampling

Client sampling refers to the process of selecting a subset of available clients to participate in a given round of federated training. Instead of involving all clients, the server chooses a smaller group to perform local training and send updates at each epoch.

The most common approach to client sampling is random sampling. Let N be the total number of available clients in the system. In each communication round t , the server aims to select a subset of K clients to participate. This selection can be defined by a sampling fraction C , where $K = C \cdot N$.

The most common methodologies for selecting the K clients are:

- **Uniform Random Sampling:** The server randomly selects K clients from N available clients each round with equal probability, providing simple, unbiased implementation and guaranteed long-term participation, though proving inefficient under highly non-IID data distributions or varying client data informativeness
- **Stratified Sampling:** Clients are divided into strata based on characteristics (e.g., location, data distribution, device capabilities, network conditions), then sampled proportionally or disproportionately from each stratum, ensuring diverse representation and improved robustness in non-IID settings, though requiring prior client categorization knowledge and increased implementation complexity.
- **Utility-Based Sampling:** Clients are selected based on calculated utility scores (e.g., local loss, data diversity, prediction uncertainty) to prioritize those expected to contribute most to global model improvement, potentially accelerating convergence and enhancing model quality, though introducing computational overhead, potential bias from flawed metrics, and privacy concerns if client characteristics are exposed during scoring.

The primary factors governing client sampling include sampling rate, data distribution characteristics (sampling method selection significantly impacts performance under non-IID data conditions), participation fairness (long-term participation equity ensures all eligible clients contribute to global model development), and privacy preservation (sophisticated sampling strategies may inadvertently expose client characteristics, requiring careful privacy-preserving design).

Since the sampling rate depends on the application and context, we experiment with different sample rates, ranging from 10 – 50%, and implement both uniform and stratified sampling based on regional properties.

4.5 Hybrid Methods

We experiment with multiple hybrid approaches to optimize the communication efficiency. First, we integrate top-performing methods with quantization to reduce communication package sizes between server and clients. Second, we combine delta encoding with top-k sparsification by computing parameter deltas and applying sparsification to retain only the most significant changes, transmitting merely 0.1% to 5% of potential updates for maximum efficiency. These hybrid strategies collectively address multiple FL bottlenecks with minimal model performance loss.

5 Results

This section presents the empirical results obtained from testing the techniques described in Section 4. We provide detailed descriptions of our experimental results for top-k sparsification (Section 5.1), update frequency reduction (Section 5.2), delta encoding (Section 5.3), client sampling (Section 5.4), and hybrid method combinations (Section 5.5).

5.1 Top-k Sparsification

Figure 9 presents top-k sparsification results across k values ranging from 1% to 30% for all methods evaluated in D9.2 and detailed in Table 1. A consistent performance improvement emerges as k increases, aligning with expectations that model performance correlates with parameter count. At $k=30%$, we observe minimal performance degradation compared to full-parameter federated learning fine-tuning, establishing this as an effective upper threshold. While full model fine-tuning demonstrates the most consistent results, it remains the most computationally expensive approach. Notably, LoRA rank 16 maintains its position as the best-performing method, consistent with our original findings from D9.2. These results confirm that using only a subset of parameters per communication round – with dynamic parameter selection – can significantly reduce communication overhead while preserving model performance, offering an optimal trade-off between efficiency and accuracy in FL scenarios.

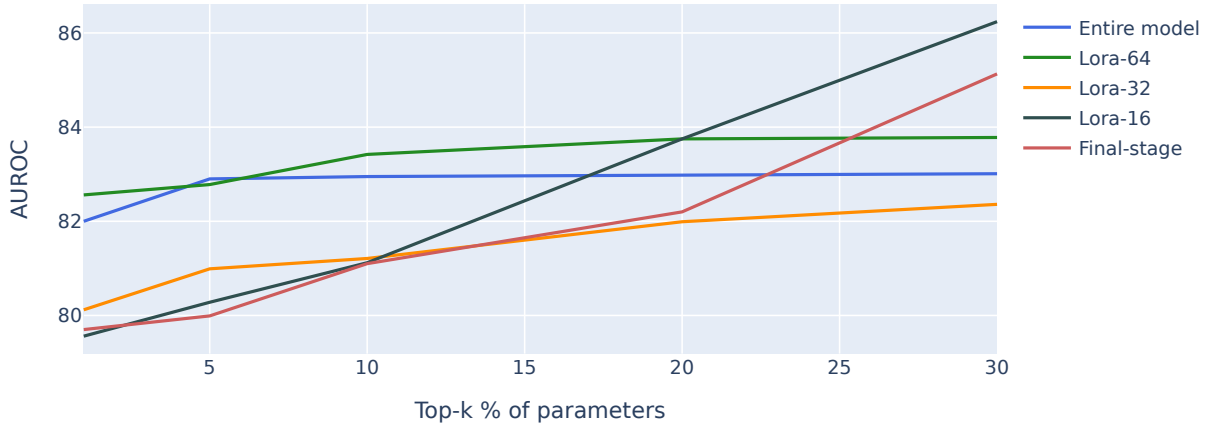


Figure 9: Performance comparison of federated fine-tuning methods across various top-k sparsification parameters applied to pre-trained foundation models for different fine-tuning strategies.

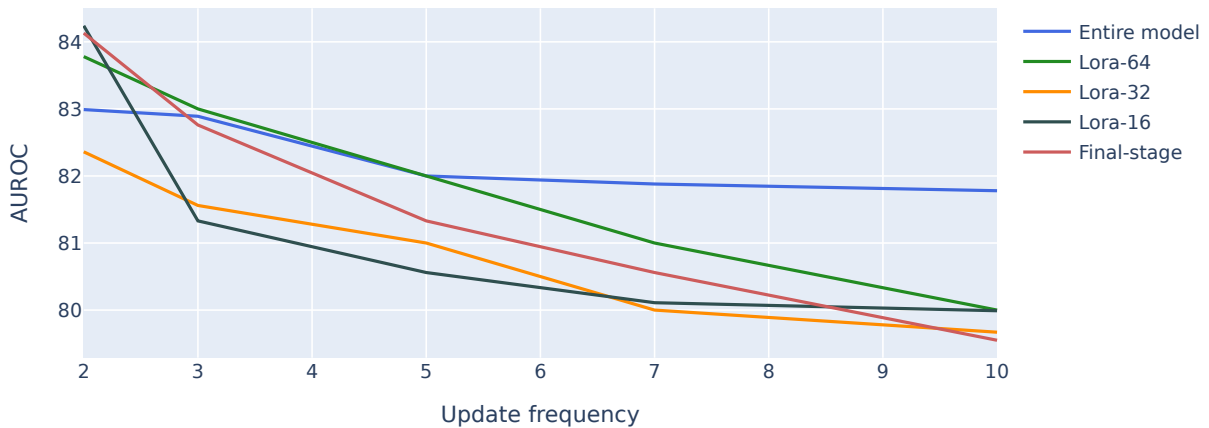


Figure 10: Performance comparison of federated fine-tuning methods across update frequencies applied to pre-trained foundation models for different fine-tuning strategies.

5.2 Update Frequency Reduction

Figure 10 illustrates update frequency reduction performance across intervals ranging from 2 to 10, with baseline results (frequency of 1) detailed in Table 1. Consistent with top-k sparsification

findings, full model fine-tuning exhibits the most stable performance, aligning with the expectation that higher parameter counts improves model robustness against communication constraints. However, unlike top-k sparsification, update frequency reduction produces more pronounced performance degradation across all methods, with particularly significant drops observed in the best-performing approaches. This deterioration stems from client drift, where reduced communication frequency allows local model updates to diverge substantially from the global model, compromising convergence quality and overall FL effectiveness.

The contrasting behavior between top-k sparsification and update frequency reduction reveals fundamental differences in their impact on FL dynamics. While top-k methods preserve communication timing but reduce information density, frequency reduction maintains full information exchange but disrupts synchronization patterns. This suggests that temporal consistency in FL for prostate cancer detection is more critical than parameter completeness for maintaining model convergence.

The disproportionate performance loss in high-performing methods (such as LoRA rank 16) indicates that these parameter-efficient techniques are more sensitive to communication disruptions. This phenomenon likely arises from their reliance on precise gradient alignment and coordinated low-rank updates across clients.

5.3 Delta Encoding

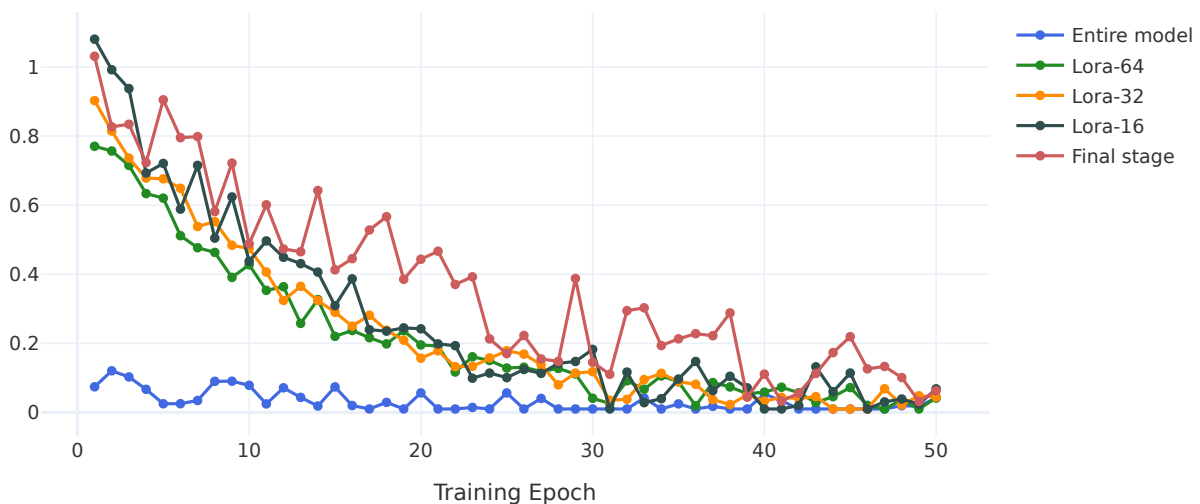


Figure 11: Ratio of non-zero (sparse) parameters over 50 training epochs.

For delta encoding, we illustrate the ratio of non-zero (sparse) parameters over 50 training epochs in Figure 11, which illustrates that the number of communicated parameters decreases progressively over time as the model specializes and converges toward optimal parameter values. This temporal sparsification occurs because parameter updates become increasingly localized and refined as training advances, with fewer parameters requiring significant modifications in later epochs.

Again, full model fine-tuning exhibits the most consistent behavior, maintaining the highest absolute number of parameters while achieving the smallest sparsity ratio due to its comprehensive parameter space. For parameter-efficient methods, we observe a more pronounced and consistent decrease in non-zero parameters, with final-stage fine-tuning displaying more dramatic sparsification patterns given its inherently smaller parameter count. This enhanced sparsity in parameter-efficient approaches reflects their concentrated optimization focus, where updates become highly targeted as the model approaches convergence.

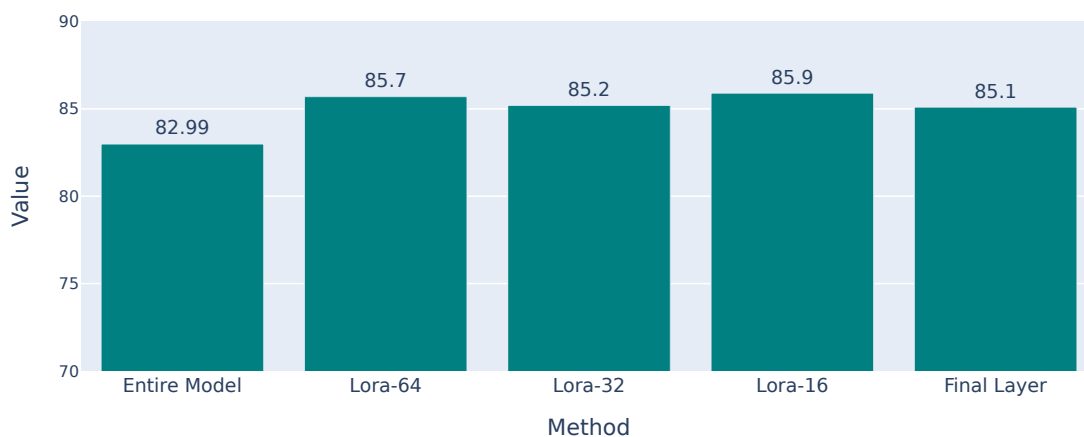


Figure 12: Performance comparison for 50 epochs training with delta encoding.

Figure 12 illustrates the final performance for all methods tested with delta encoding implementation. We observe that for certain parameter-efficient methods, such as LoRA-64 and LoRA-32, delta encoding acts as an effective regularization mechanism, improving performance beyond baseline configurations. This regularization effect emerges from the selective communication of only significant parameter changes, which filters out noise and minor fluctuations that might otherwise destabilize training, while overall maintaining competitive performance levels across all evaluated approaches.

The regularization benefits observed in higher-rank LoRA configurations suggest that delta encoding serves as an implicit gradient filtering mechanism. By transmitting only parameters that exceed the delta threshold, the method naturally suppresses small, potentially noisy updates that could lead to overfitting or unstable convergence. This effect is particularly pronounced in LoRA-64 and LoRA-32, where the larger parameter spaces are more susceptible to such noise accumulation.

Conversely, lower-rank methods (LoRA-16) show more modest changes, indicating that their inherently constrained parameter spaces already provide sufficient regularization.

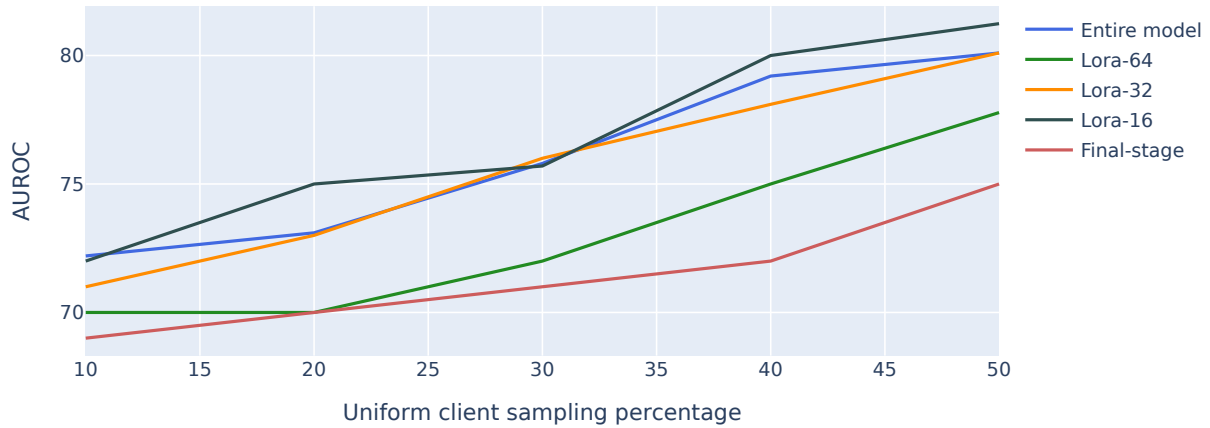


Figure 13: Performance comparison of federated fine-tuning methods under uniform client sampling strategies with varying participation rates, evaluated across pre-trained foundation models for different fine-tuning strategies.

5.4 Client Sampling

We evaluate two distinct sampling strategies: uniform sampling with varying percentages of patient collections per round, and stratified sampling employing proportionate selection from regionally-grouped collections to ensure geographical representation.

Figure 13 presents the performance outcomes for uniformly sampled collections. In contrast to previously examined communication optimization techniques, client sampling introduces the most substantial performance degradation, with even generous sample sizes of 50% resulting in performance decreases of at least 10% across all evaluated methods.

This pronounced sensitivity to client sampling reveals that the performance for the running use-case is dependent on comprehensive client participation rather than communication efficiency alone. The substantial degradation even at moderate sampling rates suggests that the heterogeneity and unique data distributions across clients cannot be adequately captured through subset sampling, regardless of the underlying parameter-efficient method employed.

Figure 14 presents the performance outcomes for stratified sampling collections based on regional groupings. We observe that the patterns identified in uniform sampling scenarios are preserved under stratified sampling, with performance degradation remaining substantial despite the improved representativeness of geographical distribution. This consistency indicates that sampling strategy refinements cannot compensate for the inherent information loss associated with reduced client participation.

These findings establish client sampling as a least suitable optimization strategy and should only be considered when computational or coordination constraints absolutely prevent full client participation. The results emphasize the importance of maximizing client inclusion in

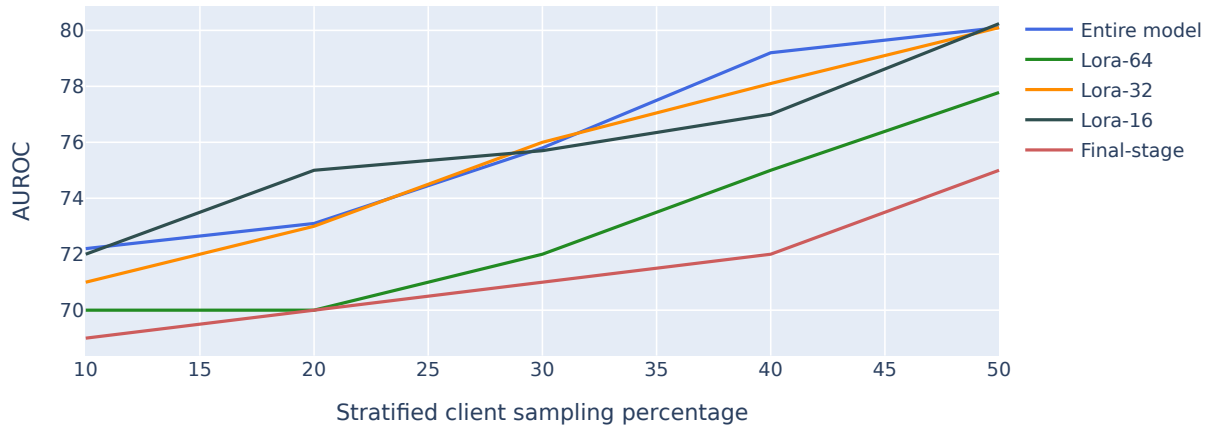


Figure 14: Performance comparison of federated fine-tuning methods under stratified client sampling strategies with varying participation rates, evaluated across pre-trained foundation models for different fine-tuning strategies.

the FL network, even at the cost of increased communication overhead or extended training duration.

5.5 Hybrid Methods

Method	32b AUROC (%)	4b AUROC (%)
Top-k (LoRA-16)	86.24 (0.023)	85.12 (0.025)
Freq. Reduction (LoRA-16)	84.24 (0.024)	84.1 (0.024)
Delta Encoding (LoRA-64)	85.7 (0.024)	85.09 (0.026)
Client Sampling (Uniform, LoRA-16)	81.24 (0.025)	79.14 (0.025)

Table 2: Performance and confidence intervals comparison of best performing compression techniques and their quantized counterparts.

Table 2 presents the performance evaluation of the best-performing efficient fine-tuning methods when implemented with 4-bit quantized model variants. Consistent with the findings from D9.2, the performance degradation introduced by quantization remains small, bringing evidence that the federated fine-tuning approaches can be robust to precision reduction.

These results establish quantization as a viable optimization strategy.

Table 3 presents the performance of combining top-k sparsification with delta encoding techniques, achieving a substantial 85% reduction in parameter count while maintaining near-optimal performance levels. This hybrid approach leverages the complementary strengths

Method	AUROC (%)	% Parameters
Top-k (LoRA-16)	86.24 (0.023)	30%
Delta Encoding (LoRA-64)	85.7 (0.024)	15%
Combined (LoRA-16)	85.24 (0.025)	9%

Table 3: Performance, confidence intervals, and parameter efficiency comparison between hybrid top-k and delta encoding implementation versus independent application of individual compression techniques.

of both compression methods, where top-k selection identifies the most critical parameter updates and delta encoding efficiently represents the magnitude differences.

6 Discussion and Conclusions

The empirical results presented in Section 5 reveal insights into the effectiveness and trade-offs of distinct communication optimization strategies for federated fine-tuning in medical imaging applications. In this section we examine the key findings, their implications, and the practical considerations for deploying these techniques in real-world FL scenarios.

Performance-Efficiency Trade-offs The experimental results demonstrate a clear hierarchy in the effectiveness of communication optimization strategies. Top-k sparsification emerges as the most promising approach, achieving substantial communication reduction (up to 70% parameter reduction at $k=30\%$) with minimal performance degradation. This technique’s success stems from its ability to preserve temporal consistency in federated communication while selectively transmitting only the most significant parameter updates.

In contrast, update frequency reduction introduces more pronounced performance penalties, particularly affecting parameter-efficient methods like LoRA. The disproportionate impact on high-performing methods reveals a fundamental trade-off: while frequency reduction offers straightforward implementation and guaranteed bandwidth savings, it disrupts the synchronization patterns for federated convergence. This suggests that temporal consistency in FL is more valuable than parameter completeness for maintaining model quality.

Delta encoding provides unique advantages through its adaptive nature, with communication overhead naturally decreasing as training progresses. The observed regularization effects in higher-rank LoRA configurations (LoRA-64, LoRA-32) indicate that delta encoding serves as an implicit gradient filtering mechanism, potentially improving model generalization. However, this benefit comes at the cost of method-specific threshold tuning requirements, reducing the technique’s generalizability across different model architectures and datasets.

Method-Specific Optimization Challenges A critical limitation revealed by the analysis is the absence of universally optimal hyper-parameters across optimization techniques. Each method requires tuning of specific parameters: sparsification thresholds for top-k, frequency intervals for update reduction, and delta thresholds for encoding approaches. This creates

deployment complexity in heterogeneous federated environments where multiple model architectures or diverse client capabilities necessitate distinct optimization strategies.

Furthermore, the parameter-efficient methods exhibit varying sensitivity to different optimization approaches. While LoRA-16 consistently performs well with top-k sparsification, higher-rank LoRA configurations benefit more from delta encoding's regularization effects. This heterogeneity suggests that FL deployments may require adaptive optimization strategies that dynamically select techniques based on model architecture and training progress.

Client Participation Criticality The substantial performance degradation observed with client sampling (10% performance loss even at 50% participation rates) leads to a conclusion for the running use-case: comprehensive client participation is irreplaceable in FL. Neither uniform nor stratified sampling strategies can adequately compensate for the loss of client-specific data distributions, indicating that the value of FL lies in capturing the full spectrum of data heterogeneity rather than representative subsets.

This finding can impact FL deployment strategies. Organizations may prioritize maximizing client participation over communication efficiency, suggesting that client sampling should only be considered when computational or coordination constraints absolutely prevent full participation.

Hybrid Approach Viability The successful combination of top-k sparsification and delta encoding (achieving 85% parameter reduction with minimal performance loss) demonstrates the potential for hybrid optimization strategies. However, this success comes with increased implementation complexity and hyperparameter tuning requirements. Each component technique requires independent optimization, potentially leading to multiplicative complexity in parameter space exploration.

The hybrid approach's effectiveness suggests that complementary compression techniques can achieve superior efficiency compared to individual methods. However, the practical deployment of such systems requires sophisticated parameter management and potentially adaptive tuning mechanisms to maintain optimal performance across diverse federated scenarios.

Quantization Robustness The minimal performance impact of 4-bit quantization across all optimization techniques provides encouraging evidence for memory-efficient federated learning deployments. This robustness enables practical implementation on resource-constrained edge devices without compromising learning effectiveness. The preservation of performance under quantization suggests that critical information for federated adaptation is maintained even with reduced numerical precision.

Practical Deployment and Integration Considerations The results highlight several practical challenges for real-world medical FL deployments:

- **Method Selection Complexity:** The absence of universally optimal techniques necessitates evaluation of specific use case requirements, client capabilities, and performance priorities.

- **Hyperparameter Management:** Each optimization technique requires method-specific tuning, creating potential scalability challenges in large-scale federated networks.
- **Client Heterogeneity:** The varying sensitivity of different model architectures to optimization techniques suggests the need for adaptive or client-specific (personalized) optimization strategies.
- **Communication Infrastructure:** The trade-offs between communication frequency and information density must be carefully balanced based on network capabilities and constraints.

Future Research Directions These findings suggest several promising research directions: adaptive optimization frameworks that dynamically select techniques based on training progress and client characteristics, automated hyperparameter tuning for federated environments, and unified compression approaches that combine multiple techniques while minimizing configuration complexity. Additionally, investigating the generalizability of these findings across different medical imaging tasks and exploring client-aware optimization strategies could further advance FL effectiveness.

References

- [1] Sijia Chen, Ningxin Su, and Baochun Li. Calibre: Towards fair and accurate personalized federated learning with self-supervised learning.
- [2] Yae Jee Cho, Luyang Liu, Zheng Xu, Aldi Fahrezi, Matt Barnes, and Gauri Joshi. Heterogeneous lora for federated fine-tuning of on-device foundation models. In *International Workshop on Federated Learning in the Age of Foundation Models in Conjunction with NeurIPS 2023*, 2023.
- [3] Yiyuan Yang, Guodong Long, Tao Shen, Jing Jiang, and Michael Blumenstein. Dual-personalizing adapter for federated foundation models. *arXiv:2403.19211*, 2024.
- [4] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv:2108.07258*, 2021.
- [5] Yuting He, Fuxiang Huang, Xinrui Jiang, Yuxiang Nie, Minghao Wang, Jiguang Wang, and Hao Chen. Foundation model for advancing healthcare: challenges, opportunities and future directions. *IEEE Reviews in Biomedical Engineering*, 2024.
- [6] Weiming Zhuang, Chen Chen, and Lingjuan Lyu. When foundation model meets federated learning: Motivations, challenges, and future directions. *arXiv:2306.15546*, 2023.
- [7] Shaoting Zhang and Dimitris Metaxas. On the challenges and perspectives of foundation models for medical image analysis. *Medical image analysis*, 91:102996, 2024.
- [8] Mahdi Beitollahi, Mingrui Liu, and Ning Lu. Dsfl: Dynamic sparsification for federated learning. In *2022 5th International Conference on Communications, Signal Processing, and their Applications (ICCSPA)*, pages 1–6. IEEE, 2022.
- [9] Kangkang Sun, Hansong Xu, Kun Hua, Xi Lin, Gaolei Li, Tigang Jiang, and Jianhua Li. Joint top-k sparsification and shuffle model for communication-privacy-accuracy tradeoffs in federated-learning-based iov. *IEEE Internet of Things Journal*, 11(11):19721–19735, 2024.
- [10] Yang Xu, Yunming Liao, Hongli Xu, Zhenguo Ma, Lun Wang, and Jianchun Liu. Adaptive control of local updating and model compression for efficient federated learning. *IEEE Transactions on Mobile Computing*, 22(10):5675–5689, 2022.
- [11] Felix Sattler, Arturo Marban, Roman Rischke, and Wojciech Samek. Cfd: Communication-efficient federated distillation via soft-label quantization and delta coding. *IEEE Transactions on Network Science and Engineering*, 9(4):2025–2038, 2021.
- [12] Amirhossein Malekijoo, Mohammad Javad Fadaeieslam, Hanieh Malekijou, Morteza Homayounfar, Farshid Alizadeh-Shabdiz, and Reza Rawassizadeh. Fedzip: A compression framework for communication-efficient federated learning. *arXiv:2102.01593*, 2021.
- [13] GOPAL Patro and Kishore Kumar Sahu. Normalization: A preprocessing stage. *arXiv preprint arXiv:1503.06462*, 2015.
- [14] Christian Roest, TC Kwee, A Saha, JJ Fütterer, Derya Yakar, and H Huisman. Ai-assisted

- biparametric mri surveillance of prostate cancer: feasibility study. *European radiology*, 33(1):89–96, 2023.
- [15] Zhenda Xie, Zheng Zhang, Yue Cao, Yutong Lin, Jianmin Bao, Zhuliang Yao, Qi Dai, and Han Hu. Simmim: A simple framework for masked image modeling. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9653–9663, 2022.
- [16] Zekai Chen, Devansh Agarwal, Kshitij Aggarwal, Wiem Safta, Mariann Micsinai Balan, and Kevin Brown. Masked image modeling advances 3d medical image analysis. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1970–1980, 2023.
- [17] Jeong Hoon Lee, Cynthia Xinran Li, Hassan Jahanandish, Indrani Bhattacharya, Sulaiman Vesal, Lichun Zhang, Shengtian Sang, Moon Hyung Choi, Simon John Christoph Soerensen, Steve Ran Zhou, et al. Prostate-specific foundation models for enhanced detection of clinically significant cancer. *arXiv e-prints*, pages arXiv–2502, 2025.
- [18] Matthew McDermott, Haoran Zhang, Lasse Hansen, Giovanni Angelotti, and Jack Galifant. A closer look at auROC and aupRC under class imbalance. *Advances in Neural Information Processing Systems*, 37:44102–44163, 2024.
- [19] Usha Ruby, Vamsidhar Yendapalli, et al. Binary cross entropy with deep learning technique for image classification. *Int. J. Adv. Trends Comput. Sci. Eng*, 9(10), 2020.
- [20] Zhenxun Zhuang, Mingrui Liu, Ashok Cutkosky, and Francesco Orabona. Understanding adamw through proximal methods and scale-freeness. *arXiv preprint arXiv:2202.00089*, 2022.
- [21] Robert G Newcombe. Confidence intervals for an effect size measure based on the mann–whitney statistic. part 1: general issues and tail-area-based methods. *Statistics in medicine*, 25(4):543–557, 2006.

Part II

Communication-Efficient Federated learning: New algorithms for Gradient Projection onto Historical Descent Directions

1 Introduction

In recent years, Federated Learning (FL) [KMA⁺21] has emerged as a promising paradigm for training Machine Learning (ML) models across distributed data owners, without requiring direct access to the underlying data. This framework is particularly attractive in a landscape where organizations are increasingly concerned about data privacy, regulatory constraints (GDPR, AI Act), and the rising costs of centralized data infrastructure. FL is a setting where multiple entities, called clients, collaborate in solving a ML problem, under the coordination of a central server. Mathematically, various ML tasks can be formulated as the optimization problem:

$$\text{Minimize } f(w) = \frac{1}{M} \sum_{i=1}^M f_i(w) \text{ over } \mathbb{R}^d, \quad (1)$$

where the vector w represents the parameters of a statistical model and $f_i : \mathbb{R}^d \rightarrow \mathbb{R}$ is the local objective function associated with client $i \in [M]$. Typically, in ML, the functions f_i take the form

$$f_i(w) = \mathbb{E}_{(x,y) \sim \pi_i} [\mathbb{L}(y, h_w(x))], \quad (2)$$

where, in a supervised learning task, $h_w : \mathcal{X} \rightarrow \mathbb{R}$ is the prediction function (e.g., a neural network), and $\mathbb{L} : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}_+$ a loss function. The data distribution π_i is the underlying data distribution of client i , and is typically unknown. In practice, π_i may be defined as a discrete distribution over the local dataset of client i , leading to empirical risk minimization. This motivates the use of either deterministic gradient descent (GD) or, more commonly, to stochastic gradient descent (SGD), which is preferred for its computational efficiency.

The communication cost is a major bottleneck in FL, especially for optimization of models with many parameters. Two main strategies have been explored to mitigate this issue:

1. **Update frequency reduction (Part 1)**, which consists of reducing the communication frequency. Clients perform multiple local updates before sending their weights to the server for averaging.
2. **Gradient compression**, where compressed information is sent instead of full-dimensional gradients, similar to hybrid methods based on quantization, as tested in Part 1.

The first strategy involves performing multiple local updates (e.g., several SGD steps) before each communication round with the server, thereby reducing communication frequency. This

classical approach, known as FedAvg [MMR⁺17], has inspired several extensions, such as SCAFFOLD [KKM⁺20] and FedPAGE [ZLR21]. The second strategy can be broadly divided into two subcategories. The first focuses on designing and analyzing various types of compression operators (see, e.g., [XHA⁺21, Table 1] and [PD24]). The second aims to develop algorithms that remain effective under general compression schemes—or at least under broad classes of them. For example, several methods incorporate mechanisms to track and correct the compression error [KRSJ19, RSF21]. In this work, we focus on the second strategy and propose a new algorithmic approach for gradient compression in FL.

The contributions of the second part of the report can be summarized as follows:

- **Algorithm Design:** We propose ProjFL (Algorithm 1), a novel FL method that projects gradients onto a shared client-server subspace. This significantly improves the convergence speed at the cost of transmitting only one additional scalar per iteration. We also propose an Error Feedback extension (Algorithm 2) for improved performance under biased compression.
- **Theoretical Guarantees:** Under standard assumptions, we establish: (i) Linear convergence to a noise ball for SGD (Theorem 4.3); and (ii) a $O(1/T)$ rate for GD (Theorem 4.4).
- **Empirical Validation:** We evaluate our method on large-scale neural networks (LeNet-5, ResNet-20), representative of highly non-convex optimization problems. Our approach outperforms state-of-the-art baselines in terms of final accuracy and stability: it converges quickly without requiring careful hyperparameter tuning. Furthermore, it achieves up to a 8× reduction in communication cost compared to existing methods.

Outline. The remainder of this part of the report is organized as follows. Section 2 introduces the formal problem setting and reviews related work. Section 3 presents our proposed methods, which are theoretically analyzed in Section 4. Numerical experiments are reported in Section 5. We discuss the limitations of our work in Section 6, and conclude with future research directions in Section 7.

2 Setting and State of the Art

To solve the optimization problem (1) under communication constraints, a classical algorithm is FedAvg with compression (see Algorithm 3 in Appendix A), which can be written as:

$$w_{t+1} = w_t - \frac{\eta}{M} \sum_{i=1}^M \mathcal{C}(g_t^i), \quad (3)$$

where g_t^i is the stochastic gradient computed by client i at iteration $t \in \mathbb{N}$, and $\mathcal{C} : \mathbb{R}^d \rightarrow \mathbb{R}^d$ is a compression operator. Each g_t^i is assumed to be an unbiased estimator of the true gradient, i.e., $\mathbb{E}[g_t^i | w_t] = \nabla f_i(w_t)$.

For ease of exposition and analysis, we treat \mathcal{C} as a mapping from \mathbb{R}^d to \mathbb{R}^d , although in practical implementations: (i) the model parameters are typically updated layer-wise; (ii)

several compression techniques require adaptations to the standard update rule (3). We now review commonly used classes of compressors.

2.1 Compressors

Quantization-based Compressors. These methods reduce communication by lowering the bit precision of the transmitted gradient values. Two common approaches are:

- **Precision reduction:** Directly reducing the bitwidth of floating-point values. For instance, 8-bit quantization [Det16] or even 1-bit representations [SFD⁺14] are commonly used.
- **Dictionary-based methods:** Gradients are quantized via a shared codebook. For example, QSGD [AGL⁺17] defines the compressed vector $\mathcal{C}(g)$ componentwise as:

$$\mathcal{C}(g)_j = \|g\|_2 \cdot \text{sgn}(g_j) \cdot \xi_j(g, s), \quad j \in [d], \quad (4)$$

where s is the number of quantization levels, and the random variable $\xi_j(g, s)$ satisfies

$$\xi_j(g, s) = \begin{cases} \frac{\ell}{s} & \text{with probability } 1 + \ell - \frac{|g_j|}{\|g\|_2} s, \\ \frac{\ell+1}{s} & \text{otherwise,} \end{cases}$$

with $\ell \in \mathbb{N}$ chosen so that $\frac{|g_j|}{\|g\|_2} \in \left[\frac{\ell}{s}, \frac{\ell+1}{s}\right]$.

In this scheme, clients send the tuple $(\|g\|_2, \sigma, \zeta)$, where σ contains the signs of g and ζ the quantized coefficients.

For additional examples, including low-rank quantization methods, we refer the reader to [XHA⁺21, PD24].

Sparsification-based Compressors. These methods reduce communication by enforcing sparsity, *i.e.*, zeroing out a large fraction of the gradient coordinates:

- **Rand- k :** Uniformly selects k coordinates to retain:

$$\mathcal{C}(g) = \frac{d}{k} \sum_{j \in S} g_j e_j,$$

where S is a random subset of $[d]$ with cardinality k , and $\{e_j\}_{j=1}^d$ is the canonical basis of \mathbb{R}^d . The scaling factor d/k ensures unbiasedness.

In practice, sparsification is typically applied layer-wise, and k may be set as a fixed proportion of non-zero coordinates.

- **Top- k :** Retains the k largest-magnitude components:

$$\mathcal{C}(g) = \sum_{i=d-k+1}^d g_{(i)} e_{(i)},$$

where coordinates are sorted by absolute value: $|g_{(1)}| \leq \dots \leq |g_{(d)}|$. Variants such as Threshold- v [DBA⁺20] retain all components exceeding a fixed threshold.

At the implementation level, a sparse vector is typically represented using two components: one vector containing the values of the selected elements of g , and another containing the ordered indices of the nonzero elements of $\mathcal{C}(g)$. Quantization and sparsification methods can be combined: for example, one can first apply a **Top- k** compressor and then quantize the k nonzero components to reduce their precision.

In the context of analyzing the convergence of (3), it is useful to distinguish compressors based on their bias properties. A compressor $\mathcal{C} : \mathbb{R}^d \rightarrow \mathbb{R}^d$ is said to be unbiased if for all $g \in \mathbb{R}^d$, $\mathbb{E}[\mathcal{C}(g)] = g$ (see Assumption A1 for details); otherwise, it is biased. Examples of unbiased compressors include the dictionary-based method defined in (4) and **Rand- k** , whereas **Top- k** is biased. We refer to [KFJ18, SSR21] for comprehensive discussions on unbiased compressors.

Unbiased compressors enjoy appealing theoretical guarantees: for example, in gradient descent, one can establish convergence to a neighborhood of the optimum under unbiased compression when optimizing convex objectives [KFJ18]. In contrast, analyzing biased compressors is more challenging. In fact, it has been shown that no general convergence guarantees can be obtained for biased compression, even in the strongly convex setting [BHRS23, Section 5.2]. Nevertheless, in practice, biased compressors such as **Top- k** or its variants often outperform unbiased, randomized alternatives [DMJVE16, AH17]. To better understand and stabilize these biased methods, several new algorithmic frameworks have been proposed, which we now review.

2.2 Algorithm Design

A widely adopted mechanism in communication-efficient optimization is Error Feedback (EF), where each client stores the accumulated compression error and reinjects it in the next update [SFD⁺14, KRSJ19]. This approach enables convergence guarantees even when using biased compressors, particularly for strongly convex objectives [SCJ18, AHJ⁺18, BHRS23]. Moreover, EF can also be applied on the server side, especially in cross-device FL scenarios [TYL⁺19].

To improve theoretical guarantees and stability, the EF21 method was proposed in [RSF21]. Rather than applying compression directly to gradients, EF21 compresses the difference between the current stochastic gradient and the previous descent direction. This design has inspired a series of extensions [FSG⁺21, MGI⁺22, GTR23, FTR23].

Another influential approach is DIANA [MGTA24], which introduces a shared memory vector maintained by both clients and the server. Compression is applied to the deviation between this memory and the local stochastic gradient. While the original method includes a proximal step to address regularized problems, a simplified version without this step can be used in the absence of regularization. Further developments and analyses around DIANA can be found in [HKM⁺23, CR22, GHR20].

3 Two New Algorithms

In this section, we introduce the algorithms evaluated in this work, specifically designed for FL under gradient compression.

3.1 Our Main Algorithm: ProjFL

Our first contribution is the ProjFL algorithm, which we now describe in detail. This algorithm leverages the fact that both the server and each client i have access to the local descent direction D_t^i at iteration t . At the next iteration, instead of compressing the full stochastic gradient g_{t+1}^i , client i projects this gradient onto the one-dimensional subspace generated by \bar{D}_t^i , the average of the previous descent directions D_t^i . This yields the decomposition $g_{t+1}^i = \alpha_{t+1}^i \bar{D}_t^i + (g_{t+1}^i)^\perp$, where $(g_{t+1}^i)^\perp$ is orthogonal to \bar{D}_t^i , i.e., $\bar{D}_t^i \cdot (g_{t+1}^i)^\perp = 0$ (see line 6 of Algorithm 1). The scalar α_{t+1}^i thus captures the entire component of the gradient in the known direction, enabling a highly compact representation. Instead of compressing the full gradient, the client compresses only the orthogonal component $(g_{t+1}^i)^\perp$ and transmits the pair $(\alpha_{t+1}^i, M_{t+1}^i)$ to the server, where M_{t+1}^i is the compressed version of $(g_{t+1}^i)^\perp$.

Two extreme cases help illustrate the benefits of this approach:

- If g_{t+1}^i is nearly aligned with \bar{D}_t^i , then α_{t+1}^i captures nearly all the gradient information, and the compressed component is negligible.
- Conversely, if g_{t+1}^i is orthogonal to \bar{D}_t^i , then $\alpha_{t+1}^i = 0$, and only the orthogonal part is transmitted, avoiding any misleading bias from projecting in the wrong direction.

Figure 15 illustrates the key differences between Algorithm 1 and EF21.

Algorithm 1 ProjFL

- 1: **Initialization:** $w_0 \in \mathbb{R}^d$, $D_0^i = 0$. Number of previous directions of descent to consider $K \in \mathbb{N}^*$.
 - 2: **for** $t = 0, \dots, T$ **do**
 - 3: **for** Each client i **do**
 - 4: Receive w_t .
 - 5: Compute Stochastic Gradient g_{t+1}^i .
 - 6: $\bar{D}_t^i = \frac{1}{K} \sum_{k=0}^{K-1} D_{t-k}^i$ \triangleright with obvious adaptation¹ when $t < K - 1$
 - 7: $\alpha_{t+1}^i = \frac{g_{t+1}^i \cdot \bar{D}_t^i}{\|\bar{D}_t^i\|_2^2}$ such that $g_{t+1}^i = \alpha_{t+1}^i \bar{D}_t^i + (g_{t+1}^i)^\perp$ satisfying $\bar{D}_t^i \cdot (g_{t+1}^i)^\perp = 0$.
 - 8: $M_{t+1}^i = \mathcal{C}((g_{t+1}^i)^\perp)$
 - 9: $D_{t+1}^i = \alpha_{t+1}^i D_t^i + M_{t+1}^i$ \triangleright Update the descent direction
 - 10: Send $(\alpha_{t+1}^i, M_{t+1}^i)$ to the Central Server.
 - 11: **Central Server:**
 - 12: $\bar{D}_t^i = \frac{1}{K} \sum_{k=0}^{K-1} D_{t-k}^i$ \triangleright Same computation as line 6
 - 13: $D_{t+1}^i = \alpha_{t+1}^i D_t^i + M_{t+1}^i$ \triangleright Update the descent direction
 - 14: $w_{t+1} = w_t - \frac{\eta}{M} \sum_{i=1}^M D_{t+1}^i$.
-

3.2 Integration of Error Feedback Mechanism

We now present an extension of Algorithm 1 that incorporates the Error Feedback mechanism. In Algorithm 2, we modify Algorithm 1 to explicitly account for the compression error. More

¹If $t < K - 1$, then \bar{D}_t^i is defined as $\bar{D}_t^i = \frac{1}{t+1} \sum_{k=0}^t D_{t-k}^i$.

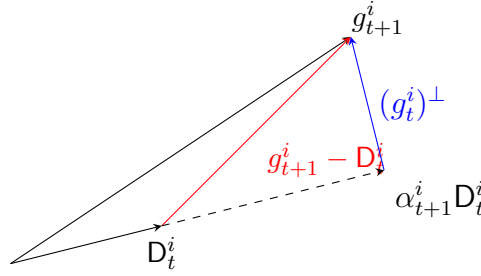


Figure 15: Comparison between EF21 and ProjFL: In EF21, client i sends the compressed difference $\mathcal{C}(g_{t+1}^i - D_t^i)$, while in ProjFL, the message is $\mathcal{C}((g_{t+1}^i)^\perp)$.

precisely, after computing the decomposition $g_{t+1}^i = \alpha_{t+1}^i \bar{D}_t^i + (g_{t+1}^i)^\perp$, each client adds the previous compression error e_t^i to the orthogonal component. The message sent to the server is then $(\alpha_{t+1}^i, \mathcal{C}((g_{t+1}^i)^\perp + e_t^i))$.

Algorithm 2 ProjFL + EF

- 1: **Initialization:** $w_0 \in \mathbb{R}^d$, $D_0^i = e_0^i = 0$, $K \in \mathbb{N}^*$.
 - 2: **for** $t = 0, \dots, T$ **do**
 - 3: **for each client** i **do**
 - 4: Receive w_t .
 - 5: Compute Stochastic Gradient g_{t+1}^i .
 - 6: $\bar{D}_t^i = \frac{1}{K} \sum_{k=0}^{K-1} D_{t-k}^i$ ▷ with obvious adaptation when $t < K - 1$ (see Alg. 1)
 - 7: $\alpha_{t+1}^i = \frac{g_{t+1}^i \cdot \bar{D}_t^i}{\|\bar{D}_t^i\|_2^2}$ such that $g_{t+1}^i = \alpha_{t+1}^i \bar{D}_t^i + (g_{t+1}^i)^\perp$ satisfying $\bar{D}_t^i \cdot (g_{t+1}^i)^\perp = 0$.
 - 8: $\tilde{g}_{t+1}^i = \alpha_{t+1}^i \bar{D}_t^i + (g_{t+1}^i)^\perp + e_t^i$ ▷ Add the previous compression error
 - 9: $M_{t+1}^i = \mathcal{C}((g_{t+1}^i)^\perp + e_t^i)$
 - 10: $D_{t+1}^i = \alpha_{t+1}^i \bar{D}_t^i + M_{t+1}^i$ ▷ Update the descent direction
 - 11: $e_{t+1}^i = (g_{t+1}^i)^\perp + e_t^i - M_{t+1}^i$ ▷ Update compression error
 - 12: Send $(\alpha_{t+1}^i, M_{t+1}^i)$ to the Central Server.
 - 13: **Central Server:**
 - 14: $\bar{D}_t^i = \frac{1}{K} \sum_{k=0}^{K-1} D_{t-k}^i$ ▷ Same computation as line 6
 - 15: $D_{t+1}^i = \alpha_{t+1}^i \bar{D}_t^i + M_{t+1}^i$ ▷ Update the descent direction
 - 16: $w_{t+1} = w_t - \frac{\eta}{N} \sum_{i=1}^N D_{t+1}^i$.
-

4 Theoretical convergence results

We give below convergence results on ProjFL (Algorithm 1). We first introduce some definitions.

Definition 4.1 (μ -strongly convex functions). *A function $f : \mathbb{R}^d \rightarrow \mathbb{R}$ is called strongly convex if there exists $\mu > 0$ such that for all $x, y \in \mathbb{R}^d$,*

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\mu}{2} \|x - y\|^2.$$

Definition 4.2 (*L-smooth functions*). A function $f : \mathbb{R}^d \rightarrow \mathbb{R}$ is called *L-smooth* if ∇f is *L-Lipschitz continuous*, i.e., for all $x, y \in \mathbb{R}^d$,

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|.$$

We make the following assumptions:

- A1.** There exists an i.i.d. sequence of compressors $(\mathcal{C}_t^i)_{i \in [M], t \geq 1}$ where for any $i \in [M]$ and $t \geq 0$, $\mathcal{C}_{t+1}^i : \mathbb{R}^d \rightarrow \mathbb{R}$ is the compressors employed by client i at iteration t . For convenience, we simply write \mathcal{C} . We assume that for all $w \in \mathbb{R}^d$, $\mathbb{E}[\mathcal{C}(w)] = w$ and $\mathbb{E}[\|\mathcal{C}(w)\|^2] \leq \beta\|w\|^2$ for some $\beta \geq 1$.²
- A2.** There exists $a, b > 0$ such that for all $w \in \mathbb{R}^d$, $\frac{1}{M} \sum_{i=1}^M \|\nabla f_i(w)\|^2 \leq a + b\|\nabla f(w)\|^2$.
- A3.** There exists an i.i.d. sequence of random vector fields $(\xi_t^i)_{i \in [M], t \geq 1}$ such that for any $i \in [M]$ and $t \geq 0$, $g_{t+1}^i = \nabla f_i(w_t) + \xi_{t+1}^i(w_t)$ and for any $w \in \mathbb{R}^d$, $\mathbb{E}[\xi_t^i(w)] = 0$ and $\mathbb{E}[\|\xi_t^i(w)\|^2] \leq \sigma^2$ for some $\sigma > 0$. Moreover, the sequences $(\xi_t^i)_{i \in [M], t \geq 1}$ and $(\mathcal{C}_t^i)_{i \in [M], t \geq 1}$ are independent.

We now discuss the role and implications of these assumptions.

- Assumption **A1** restricts our analysis to *unbiased* compression operators. Crucially, since Algorithm 1 lacks an error feedback mechanism, convergence guarantees cannot be extended to biased compressors in this framework.
- Assumption **A2** formalizes the *bounded gradient dissimilarity* condition, which quantifies data heterogeneity across clients by controlling the deviation between local gradients $\nabla f_i(w)$ and their global average $\nabla f(w)$.
- Assumption **A3** bounds the variance of stochastic gradient noise arising from mini-batch sampling. Notably, this noise intensity may vary both across clients (due to differences in local loss landscapes) and across parameter regions (e.g., near optima versus high-curvature regions).

In the following theorem - whose proof is given in Appendix B - we prove a linear convergence rate for Algorithm 1 towards a neighborhood of the optimal point.

Theorem 4.3. Assume **A1-A2-A3** and that each f_i is μ -strongly convex and *L-smooth* (see Definitions 4.1 and 4.2). Then, for any learning rate $0 \leq \eta \leq (1 + b\frac{\beta-1}{M})^{-1} \frac{2}{\mu+L}$, the sequence $(w_t)_{t \geq 0}$ generated by Algorithm 1 satisfies, for any $t \in \mathbb{N}$,

$$\mathbb{E}[\|w_t - w^*\|^2] \leq \left(1 - 2\eta \frac{\mu L}{\mu + L}\right)^t \mathbb{E}[\|w_0 - w^*\|^2] + \eta \frac{\mu + L}{2\mu L M} (a(\beta - 1) + \beta\sigma^2), \quad (5)$$

where $w^* = \arg \min_{\mathbb{R}^d} f$.

The first term on the right-hand side of (5) reflects the initial distance to the optimum. The second term is a variance term corresponding to the (squared) radius of the neighborhood around the optimal point to which the algorithm converges. This result also provides theoretical support for the common practice of decreasing the learning rate during training. A useful heuristic is as follows: at initialization, when the term $\mathbb{E}[\|w_0 - w^*\|^2]$ typically dominates,

²Note that since $\mathbb{E}[\|X - \mathbb{E}[X]\|_2^2] = \mathbb{E}[\|X\|_2^2] - \|\mathbb{E}[X]\|_2^2$, Assumption **A1** implies $\mathbb{E}[\|\mathcal{C}(w) - w\|_2^2] \leq (\beta - 1)\|w\|_2^2$.

a relatively large learning rate should be used so that the influence of this term is quickly reduced thanks to the linear convergence rate. As training progresses and the second term on the right-hand side of (5) becomes dominant, the effective "starting point" is now closer to the optimum, and using a smaller step size becomes beneficial to reduce variance and improve stability.

Our second convergence result concerns deterministic gradient descent (see the update scheme (8) in Appendix C for details). Note that in this setting, the functions f_i model different objects: they no longer represent the population loss (2), but rather the empirical loss. Consequently, training is performed using full-batch updates, which corresponds to a setting where clients hold only a small amount of data.

Theorem 4.4. *Assume A1 and that each f_i is convex and L -smooth (see Definition 4.2). Then, for any learning rate $0 < \eta < \frac{1}{\beta L}$, the sequence $(w_t)_{t \geq 0}$ generated by Algorithm 1 with deterministic gradients satisfies, for any $T \in \mathbb{N}^*$,*

$$\mathbb{E} \left[f \left(\frac{1}{T} \sum_{t=0}^{T-1} w_t \right) \right] - f^* \leq \frac{\|w_0 - w^*\|^2}{\eta T} + \frac{1}{LM \left(\frac{1}{\beta L \eta} - 1 \right)} \sum_{i=1}^M \|\nabla f_i(w^*)\|^2,$$

where $w^* \in \arg \min_{\mathbb{R}^d} f$ is any optimal point.

Note that in many applications, even if f is not globally (strongly) convex, it is so in the neighborhood of local minimizers, meaning that our results can represent the behavior of the algorithm in such regions of the search space [BCN18]. In the following section, we evaluate our methods on highly non-convex objectives.

5 Numerical experiments

In this section, we numerically evaluate the benefits of Algorithms 1 and 2 to reduce the communication cost in FL, for classification tasks using neural networks.

Datasets, models and preprocessing. We conduct experiments on MNIST and CIFAR-10 datasets³. MNIST contains 70,000 grayscale images of handwritten digits (28×28, 1 channel), split into 60,000 training and 10,000 test samples. CIFAR-10 comprises 60,000 color images (32×32, 3 channels), with 50,000 for training and 10,000 for testing, across 10 classes. We use LeNet-5 [L+15] for MNIST: two convolutional blocks (with 5×5 kernels, ReLU activations, and 2×2 max pooling), followed by two fully connected layers with ReLU. For CIFAR-10, we adopt ResNet-20 [HZRS16] implemented in [Ide]. These models are standard architectures for their respective tasks. In all settings, 20% of the training set is used for validation. Images are normalized to $[-1, 1]$, data is shuffled, and samples are evenly distributed across clients, with slight variation due to indivisibility.

Hardware and software. All experiments were conducted on an internal cluster machine equipped with an Intel Xeon Gold 5320 CPU (104 cores, 2.20 GHz), 500 GB of RAM, and a single NVIDIA A30 GPU with 24 GB of memory (driver version 545.23.08, CUDA version 12.3).

³<https://docs.pytorch.org/vision/main/datasets.html>

The software environment consisted of Python 3.9.19 and PyTorch 2.3, running on Debian GNU/Linux 12 (Bookworm). Experimental run took approximately 1 hour on average with 3 clients on 10 processes, with the most computationally intensive run requiring up to 28 hours. All experiments were performed on our institutional infrastructure on CPU; no cloud computing resources were used. Reproducing the CIFAR-10 experiments with 3 clients and a batch size of 128 requires 24 GB of RAM.

Experimental methodology. To evaluate our algorithms' ability to solve the tasks efficiently while keeping communication costs reasonable, we track convergence-related metrics—specifically the loss (cross entropy in our case), accuracy, or norm of the gradient of the loss as a function of the number of bits communicated. This part reports results based on total communication cost (uplink and downlink), while Appendix A includes extra experiments with also uplink-only evaluations and additional metrics. We focus on sparsification-based compressors and consider either $M = 3$ or $M = 10$ clients (in this part we focus on **Top- k** and consider also **Rand- k** in Appendix A).

Hyperparameter Selection. We tune the hyperparameters (early stopping criteria, batch size, and learning rate scheduler) using the update rule (3) without compression (*i.e.*, $\mathcal{C} = \text{Id}$). The goal is to reach state-of-the-art performance in terms of test cross-entropy loss. For reference, we consider values of 0.05 for MNIST and 1.4 for CIFAR-10 as representative of state-of-the-art performance for comparable architectures. Following this tuning procedure, we set the following hyperparameters:

- **Early stopping:** patience of 10 epochs and a minimum delta of 0.001.
- **Batch size:** 128 per client.
- **Learning rate scheduler:** PyTorch's `ReduceLRonPlateau`⁴ starting from 0.1, with a patience of 2 epochs, a decay factor of 0.5, and a minimum learning rate of 0.001.

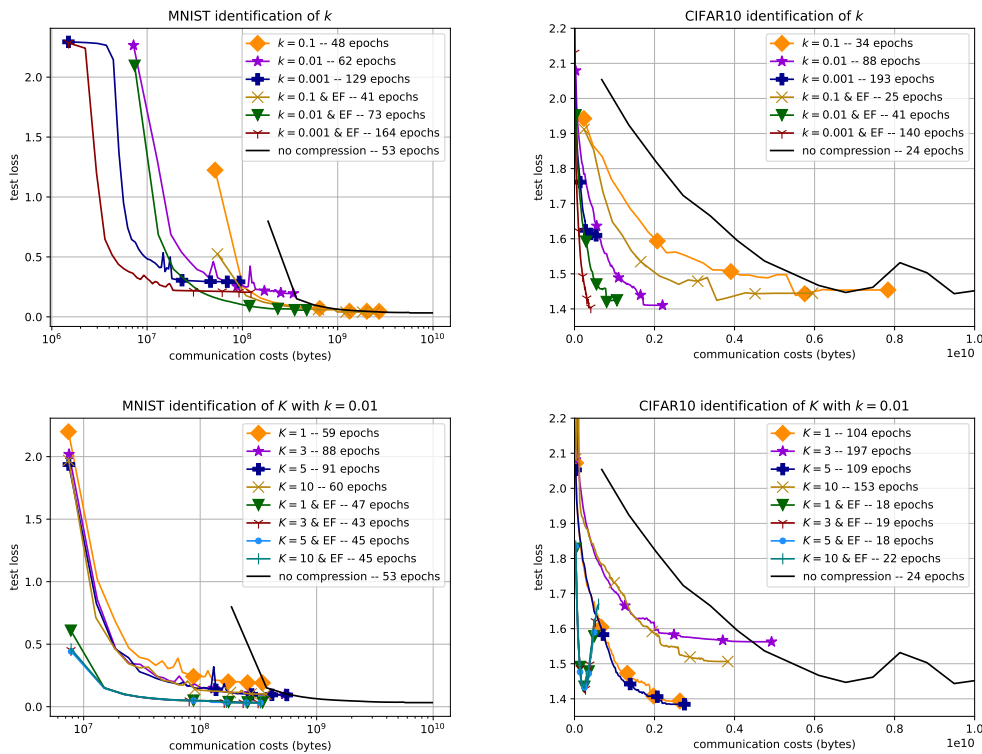
These hyperparameters are used consistently across all algorithms evaluated in our experiments.

Algorithm Parameter selection. Before comparing Algorithm 1 and Algorithm 2 to existing methods, we first need to select the parameters k (used in the **Top- k** compressor) and K (see Algorithm 1). The corresponding results are reported in Figure 16 (we use log-scale for MNIST). We also include comparisons involving EF , recalled in Algorithm 4 (see Appendix A).

Based on the observed trade-offs between convergence speed, communication cost, and stability across both datasets, the values $k = 0.01$ and $K = 3$ appear to offer a good compromise. These settings strike a strong balance between training efficiency and robustness. We note that although the algorithms differ in efficiency, they all converge to reasonable minima. Moreover, incorporating the Error Feedback mechanism consistently improves performance—an expected outcome given the use of biased compressors, as previously discussed. On the CIFAR-10 experiments for selecting K (see the two rightmost plots in Figure 16), one might be surprised by the increase in test loss at the end of training for Algorithm 2. This rise is due to overfitting,

⁴https://pytorch.org/docs/2.3/generated/torch.optim.lr_scheduler.ReduceLRonPlateau.html

Figure 16: Selection of k (**Top- k**) and K for Algorithms 1 and 2, for $M = 3$ clients.



as the training loss continues to decrease monotonically (see Appendix A). The reason the algorithm was not stopped earlier lies in the early stopping criterion (specifically the patience), which was tuned based on the behavior of the baseline FedAvg, as discussed in the previous paragraph. Naturally, had we optimized early stopping specifically for Algorithm 2, more favorable stopping conditions would have been chosen. Note that the curves start at different points, as the x-axis value of the first point corresponds to the communication cost after the first epoch.

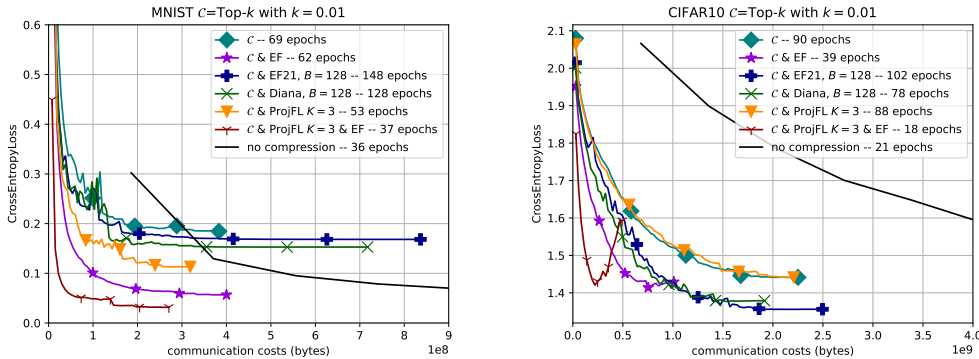
Evaluation of Algorithms 1 and 2. We compare our methods to the baselines FedAvg with compression, EF, EF21, and DIANA (see Algorithms 3, 4, 5, and 7 in Appendix A).

All algorithms are run using the same hyperparameter settings. Under our experimental setup, we found that, without further tuning, EF21 and DIANA underperformed. A similar behavior was already observed in [FTR23, Section 2.2]. To address this, we introduced an additional parameter γ into both methods, which significantly improved their performance (see Algorithms 5 and 7 for details). A thorough discussion is provided in Appendix A. The results are presented in Figure 17 (with further comparisons using different metrics and parameters in Appendix A).

On MNIST, Algorithm 1 outperforms both EF21 and DIANA. When error feedback is used, the EF algorithm remains competitive, but Algorithm 2 achieves even better performance. Notably, for similar accuracy as EF, it requires up to $8\times$ lower communication cost.

On CIFAR-10, ProjFL alone does not outperform FedAvg with compression. However, when combined with error feedback (Algorithm 2), it shows a clear advantage. Remarkably, both

Figure 17: Comparison of Algorithms 1 and 2 with FedAvg with compression, EF, EF21, and DIANA for $M = 3$ clients.



DIANA and EF21 yield very low accuracy under the same experimental conditions. To reach the same loss level as Algorithm 2, these methods require at least $3\times$ more communication cost.

We also emphasize that Algorithm 2 reaches its performance using fewer epochs, thereby reducing local computation time.

Variability across runs. To assess the stability of our method, we report the test cross-entropy loss over 10 independent runs with different random seeds, measured at the end of training. These values are provided in Appendix A. For example, for Algorithm 1, the standard deviation of the test loss is 0.0046 on MNIST and 0.016 on CIFAR-10.

6 Limitations

We outline here several limitations of our work:

- As discussed at the end of Section 4, our theoretical results are restricted to convex settings.
- Providing convergence guarantees for Algorithm 2 remains an open question, requiring technical developments that fall beyond the scope of this deliverable.
- Our approach could be combined with acceleration techniques, such as those developed in [LKQR20, HHY23], to further improve convergence. For this reason, we did not include accelerated variants such as ADIANA [LKQR20] in our comparisons.
- We evaluate our methods on only two classification datasets. While many related works focus on synthetic or simple convex tasks, we chose instead to emphasize performance on more realistic and large-scale models.
- Our experiments are conducted with a relatively small number of clients. When considering uplink communication only, this does not affect the conclusions. However, if total communication is taken into account and many clients are involved, the aggregated

updates may no longer be sparse. In such cases, compression should also be applied at the server side, as in [TYL⁺19]. Such modifications are out the scope of this work.

7 Conclusion and Discussion

In this second part of the report, we introduced two new algorithms to address the communication bottleneck in FL, in combination with gradient compression techniques. These algorithms provably improve convergence speed, while requiring only one additional scalar to be transmitted per iteration. We established convergence guarantees in both convex and strongly convex settings, under the assumption of unbiased compressors. Furthermore, we conducted extensive experiments on large-scale neural networks to assess the empirical performance of our methods beyond the theoretical setting. These algorithms consistently outperform state-of-the-art baselines. We conclude by outlining two promising directions for future work. First, our approach could be combined with acceleration techniques, such as those developed in [LKQR20, HHY23], to further improve convergence. Second, instead of projecting onto the one-dimensional subspace spanned by the average of the last K descent directions, one could consider projections onto the full K -dimensional subspace generated by these directions.

This appendix is organized as follows. In Section A, we provide implementation details and additional numerical evaluations. In Section B, we present the proof of Theorem 4.3. In Section C, we prove Theorem 4.4.

A Experimental details and Additional experiments

In this section we start by giving implementation details in Subsection A.1 and then in the following subsections provide further experiments.

A.1 Experimental details

In this subsection, we provide detailed descriptions of the algorithms evaluated in our experiments.

Algorithm 3 corresponds to the standard FedAvg algorithm with gradient compression. Algorithm 4 is a variant that incorporates the compression error using the well-known Error Feedback mechanism. All our experiments are done setting $\zeta = 0.75$. Algorithm 5 implements the EF21 algorithm from [RSF21]. In our experiments, we observed that EF21 performs poorly on large models (see the analysis provided in Subsection A.6). To address this limitation, we introduce a forgetting parameter $\gamma \in (0, 1)$, which improves the robustness of the method (see Algorithm 6).

Table 4 shows the first descent directions for Algorithms 5 and 6. This illustrates that in classical EF21, the compressed version of the initial gradients—and their propagation—persist across

Algorithm 6 EF21 with parameter $\gamma \in (0, 1)$.

- 1: **Initialization:** $D_0^i = 0 \in \mathbb{R}^d$
 - 2: **for** $t = 0, \dots, T$ **do**
 - 3: **for** Each client $i \in [M]$ **do**
 - 4: Receive w_t
 - 5: Compute Stochastic Gradient g_{t+1}^i
 - 6: $M_{t+1}^i = \mathcal{C}(g_{t+1}^i - \gamma D_t^i)$
 - 7: $D_{t+1}^i = \gamma D_t^i + M_{t+1}^i$ ▷ Update local direction of descent
 - 8: Send M_{t+1}^i to the Central Server
 - 9: **Central Server:**
 - 10: $D_{t+1}^i = \gamma D_t^i + M_{t+1}^i$
 - 11: $w_{t+1} = w_t - \frac{\eta}{M} \sum_{i=1}^M D_{t+1}^i$
-

Table 4: Comparison of the first descent direction between Algorithms 5 and 6.

Iteration k	D_k^i (Alg. 5)	D_k^i (Alg. 6)
1	$\mathcal{C}(g_1^i)$	$\mathcal{C}(g_1^i)$
2	$\mathcal{C}(g_1^i) + \mathcal{C}(g_2^i - \mathcal{C}(g_1^i))$	$\gamma \mathcal{C}(g_1^i) + \mathcal{C}(g_2^i - \gamma \mathcal{C}(g_1^i))$
3	$\mathcal{C}(g_1^i) + \mathcal{C}(g_2^i - \mathcal{C}(g_1^i))$ $+ \mathcal{C}(g_3^i - \mathcal{C}(g_1^i) - \mathcal{C}(g_2^i - \mathcal{C}(g_1^i)))$	$\gamma^2 \mathcal{C}(g_1^i) + \gamma \mathcal{C}(g_2^i - \gamma \mathcal{C}(g_1^i))$ $+ \mathcal{C}(g_3^i - \gamma^2 \mathcal{C}(g_1^i) - \gamma \mathcal{C}(g_2^i - \gamma \mathcal{C}(g_1^i)))$

Algorithm 7 DIANA [MGTa24]

- 1: **Initialization:** $h_0^i = h_0 = D_0 = 0 \in \mathbb{R}^d$
 - 2: **for** $t = 0, \dots, T$ **do**
 - 3: **for** Each client $i \in [M]$ **do**
 - 4: Receive w_t
 - 5: Compute Stochastic Gradient g_{t+1}^i
 - 6: $M_{t+1}^i = \mathcal{C}(g_{t+1}^i - h_t^i)$
 - 7: $h_{t+1}^i = h_t^i + \alpha M_{t+1}^i$ ▷ Update memory
 - 8: Send M_{t+1}^i to the Central Server
 - 9: **Central Server:**
 - 10: $M_{t+1} = \frac{1}{M} \sum_{i=1}^M M_{t+1}^i$
 - 11: $D_{t+1} = \beta D_t + h_t + M_{t+1}$ ▷ Compute the descent direction with momentum when $\beta > 0$
 - 12: $h_{t+1} = h_t + \alpha M_{t+1}$ ▷ Update memory
 - 13: $w_{t+1} = w_t - \eta D_{t+1}$
-

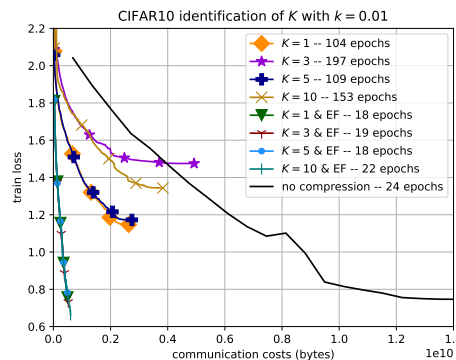
Algorithm 8 DIANA with parameter $\gamma \in (0, 1)$.

- 1: **Initialization:** $h_0^i = h_0 = D_0 = 0 \in \mathbb{R}^d$
- 2: **for** $t = 0, \dots, T$ **do**
- 3: **for** Each client $i \in [M]$ **do**
- 4: Receive w_t
- 5: Compute Stochastic Gradient g_{t+1}^i
- 6: $M_{t+1}^i = \mathcal{C}(g_{t+1}^i - \gamma h_t^i)$
- 7: $h_{t+1}^i = \gamma h_t^i + \alpha M_{t+1}^i$ ▷ Update memory
- 8: Send M_{t+1}^i to the Central Server
- 9: **Central Server:**
- 10: $M_{t+1} = \frac{1}{M} \sum_{i=1}^M M_{t+1}^i$
- 11: $D_{t+1} = \beta D_t + \gamma h_t + M_{t+1}$ ▷ Compute the descent direction with momentum when $\beta > 0$
- 12: $h_{t+1} = \gamma h_t + \alpha M_{t+1}$ ▷ Update memory
- 13: $w_{t+1} = w_t - \eta D_{t+1}$

A.2 Overfitting on the Training Set for CIFAR-10

As noticed in the main paper, in Figure 16 and 17, we can see a rising loss on the test set. This behavior is a sign of overfitting. In Figure 18 (resp. Figure 19), we reproduce the rightmost plot from Figure 16 (resp. Figure 17), but using the training dataset instead of the test dataset. As can be seen, the training loss decreases monotonically, confirming that the model continues to fit the training data while generalization performance degrades.

Figure 18: Selection of K for Algorithms 1 and 2, for $M = 3$ clients on train set.



A.3 Experiments with $M = 10$ Clients

In this subsection, we present in Figure 20 the same experiments as in Figure 17, but with $M = 10$ clients instead of 3. Both training and test losses are reported.

Figure 19: Comparison of Algorithms 1 and 2 with FedAvg with compression, EF, EF21, and DIANA for $M = 3$ clients on train set.

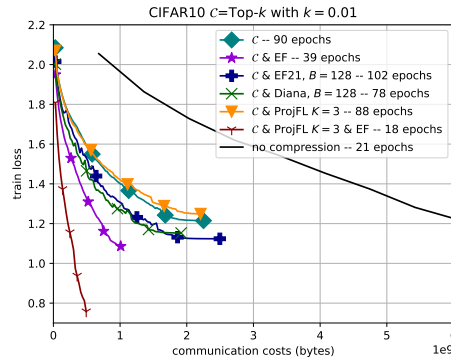
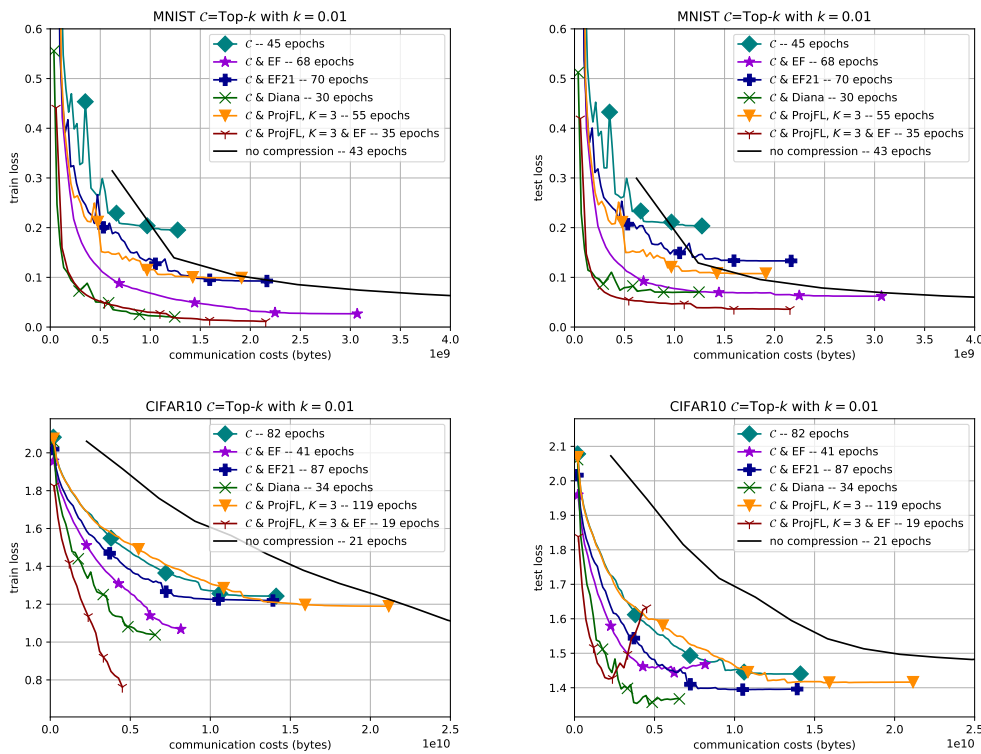


Figure 20: Comparison of Algorithms 1 and 2 with FedAvg with compression, EF, EF21, and DIANA for $M = 10$ clients.



A.4 Additional Experiments with Uplink and Downlink Communication Costs

In this subsection, we present the same experiments as in Figure 17, but considering only the uplink communication cost (Figure 21) or only the downlink communication cost (Figure 22).

Regarding the downlink communication cost, note that the central server only sends the difference $w_{t+1} - w_t$ (that is, only the values and indices of w_{t+1} that differ from w_t).

Figure 21: Uplink communication cost with $M = 3$ clients. The x-axis represents the total communication cost of the clients (*i.e.*, 3 times the communication cost per client).

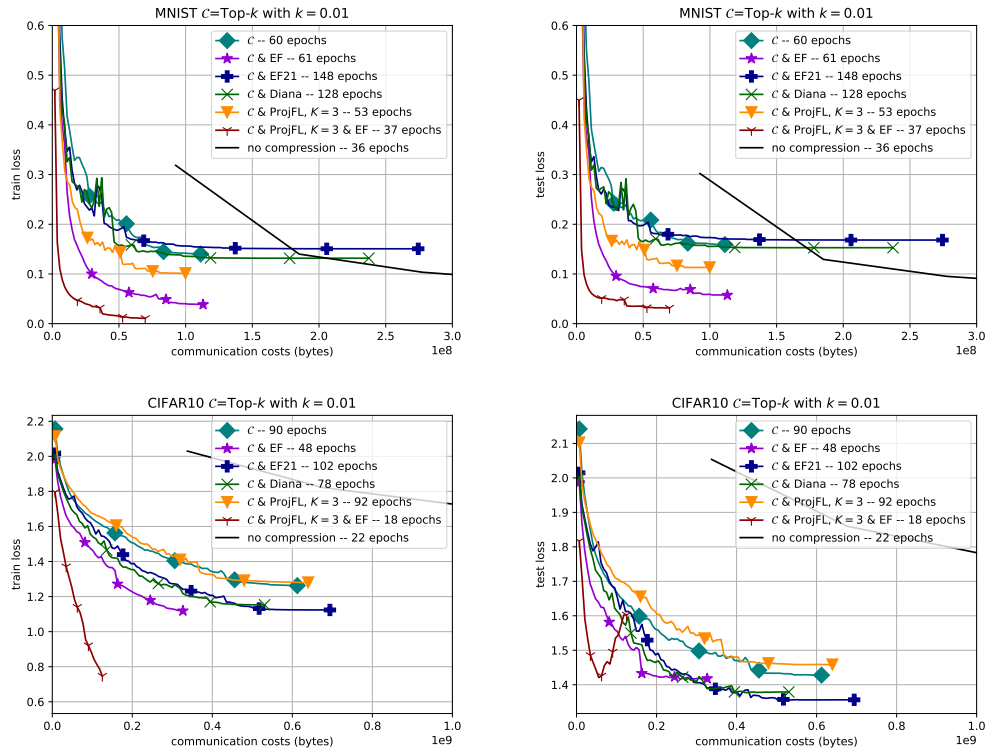
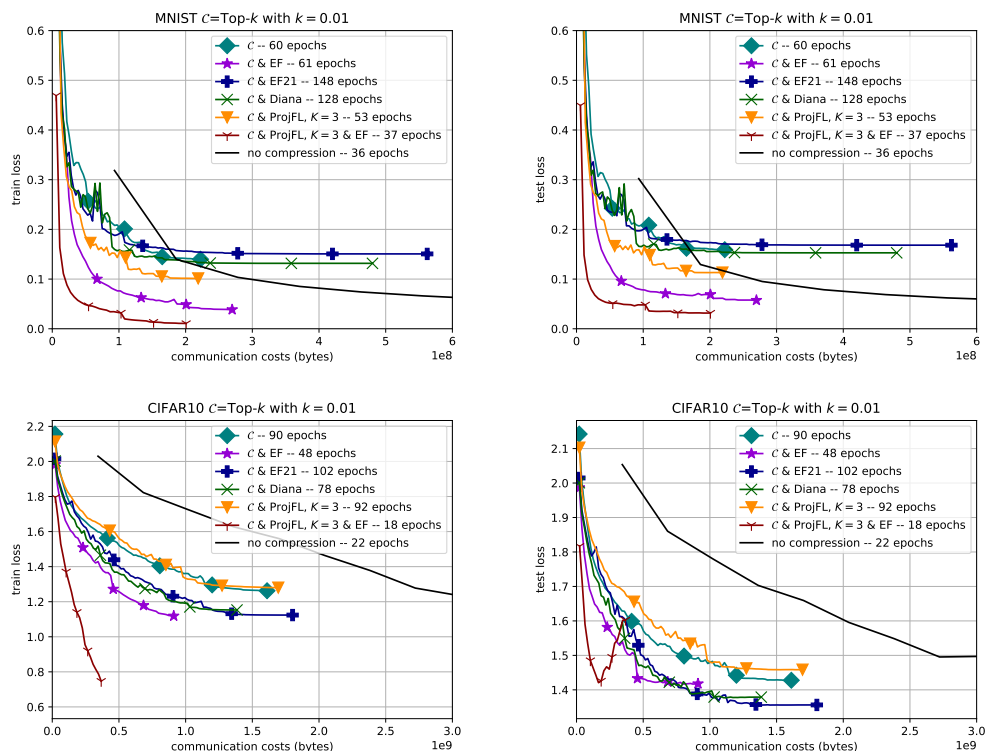


Figure 22: Downlink communication cost with $M = 3$ clients.



A.5 Accuracy

In Figure 23 (resp. Figure 24), we report both training and test accuracy for the MNIST (resp. CIFAR-10) dataset.

A.6 Evaluation of EF21 and DIANA under different hyperparameters

As already mentioned in Section 5, it was observed in [FTR23] that Algorithm 5 is particularly sensitive to the batch size. However, on our models, none of the considered batch sizes yielded favorable results, as shown in Figure 25. For this reason, all comparisons involving the EF21 method in our experiments use Algorithm 6 instead of Algorithm 5.

For DIANA, a similar observation holds. However, we also tuned the relevant hyperparameters of this algorithm, as shown in Figures 25 and 26.

A.7 Variability Across Runs

Following the same experimental conditions as described in Section 5, we report the variability of the test loss over 10 runs with different random seeds.

Table 5: Standard deviation of the test loss where $\mathcal{C}=\text{Top-0.01}$.

Algorithm	MNIST	CIFAR-10
no compression	0.0039	0.0423
Algorithm 1	0.0257	0.0284
Algorithm 2	0.0045	0.0149
Algorithm 3	0.0201	0.0211
Algorithm 4	0.0074	0.0143
Algorithm 6	0.0138	0.0221
Algorithm 8	0.0140	0.0358

We observe that Algorithm 2 has a lower standard deviation compared to the other algorithms.

B Proof of Theorem 4.3

This section is devoted to the proof of Theorem 4.3, restated below.

Theorem B.1 (Theorem 4.3). *Assume A1-A2-A3 and that each f_i is μ -strongly convex and L -smooth (see Definitions 4.1 and 4.2). Then, for any learning rate $0 \leq \eta \leq (1 + b\frac{\beta-1}{M})^{-1} \frac{2}{\mu+L}$, the sequence $(w_t)_{t \geq 0}$ generated by Algorithm 1 satisfies, for any $t \geq 0$,*

$$\mathbb{E}[\|w_t - w^*\|^2] \leq \left(1 - 2\eta \frac{\mu L}{\mu + L}\right)^t \mathbb{E}[\|w_0 - w^*\|^2] + \eta \frac{\mu + L}{2\mu LM} (a(\beta - 1) + \beta\sigma^2),$$

where $w^* = \arg \min_{\mathbb{R}^d} f$.

Figure 23: Accuracy on MNIST dataset with $M = 3$ clients. **First line:** Total communication cost. **Second line:** Uplink communication cost only. **Third line:** Downlink communication cost only.

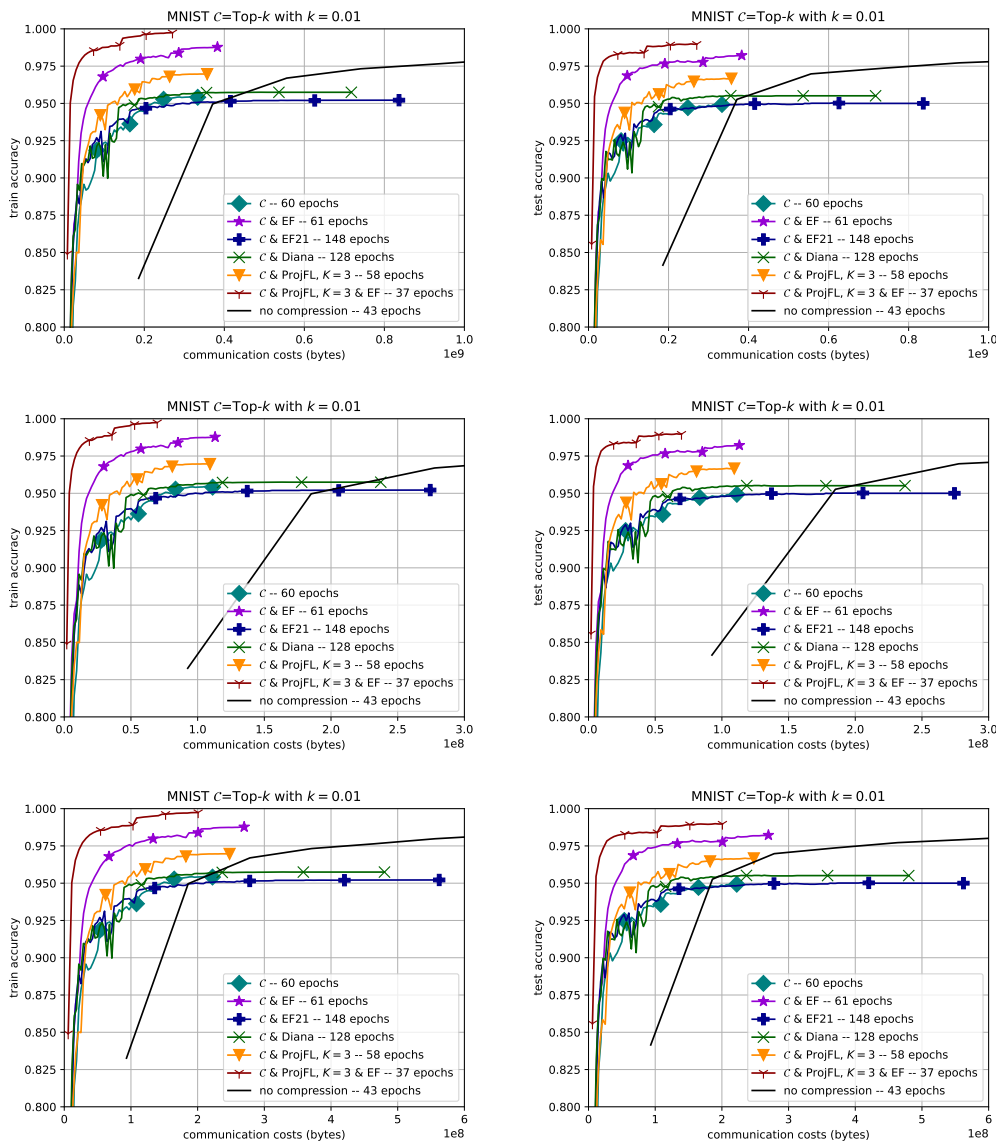


Figure 24: Accuracy on CIFAR-10 dataset with $M = 3$ clients. **First line:** Total communication cost. **Second line:** Uplink communication cost only. **Third line:** Downlink communication cost only.

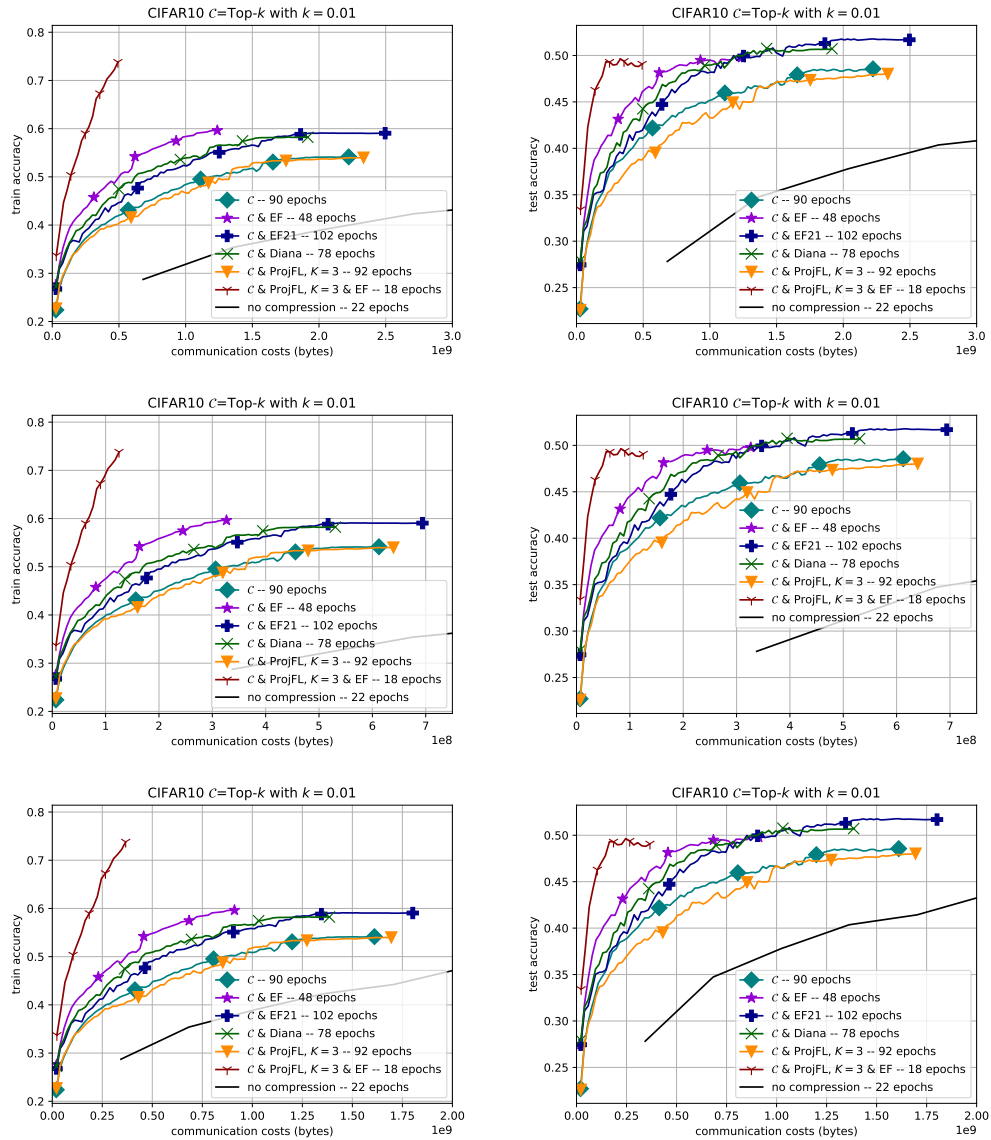


Figure 25: EF 21 (Alg. 5) under different batch sizes \mathcal{B} with $M = 3$ clients.

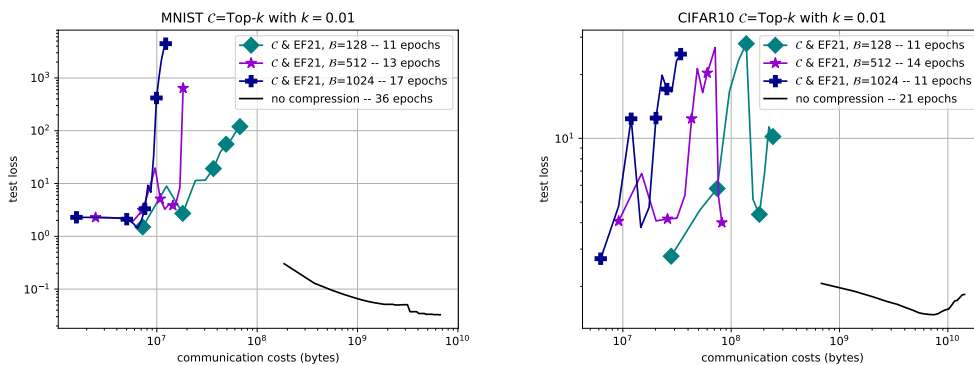


Figure 26: DIANA (Alg. 7) on MNIST under different batch sizes \mathcal{B} and hyperparameters, with $M = 3$ clients.

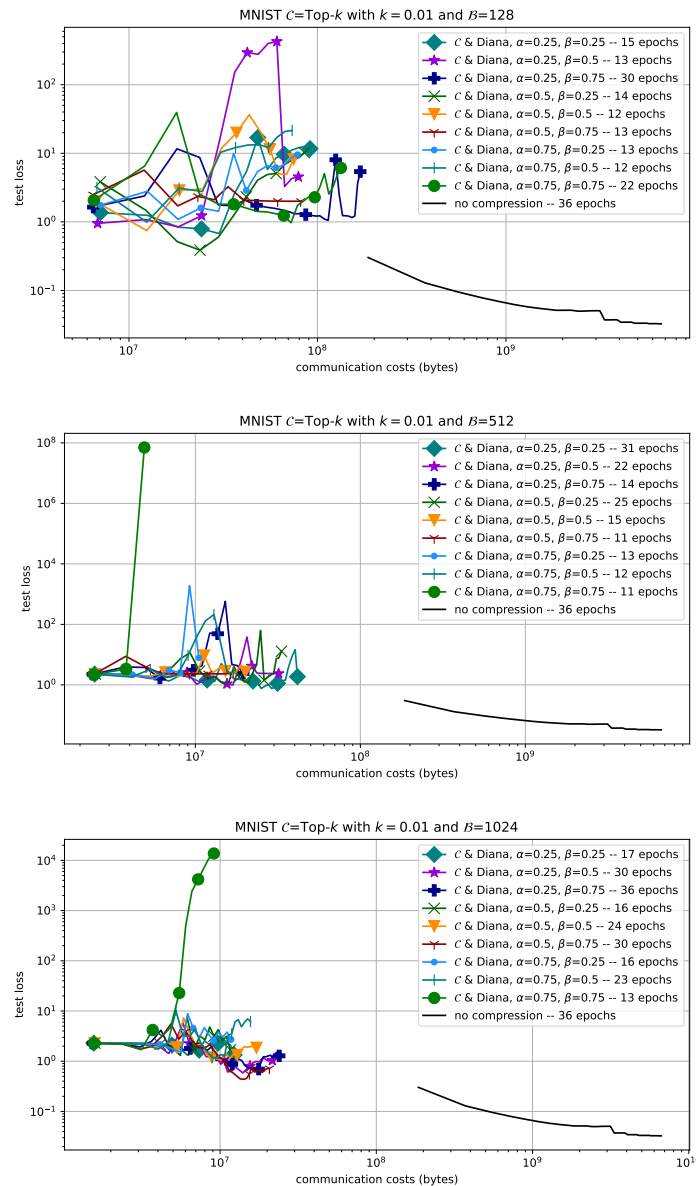
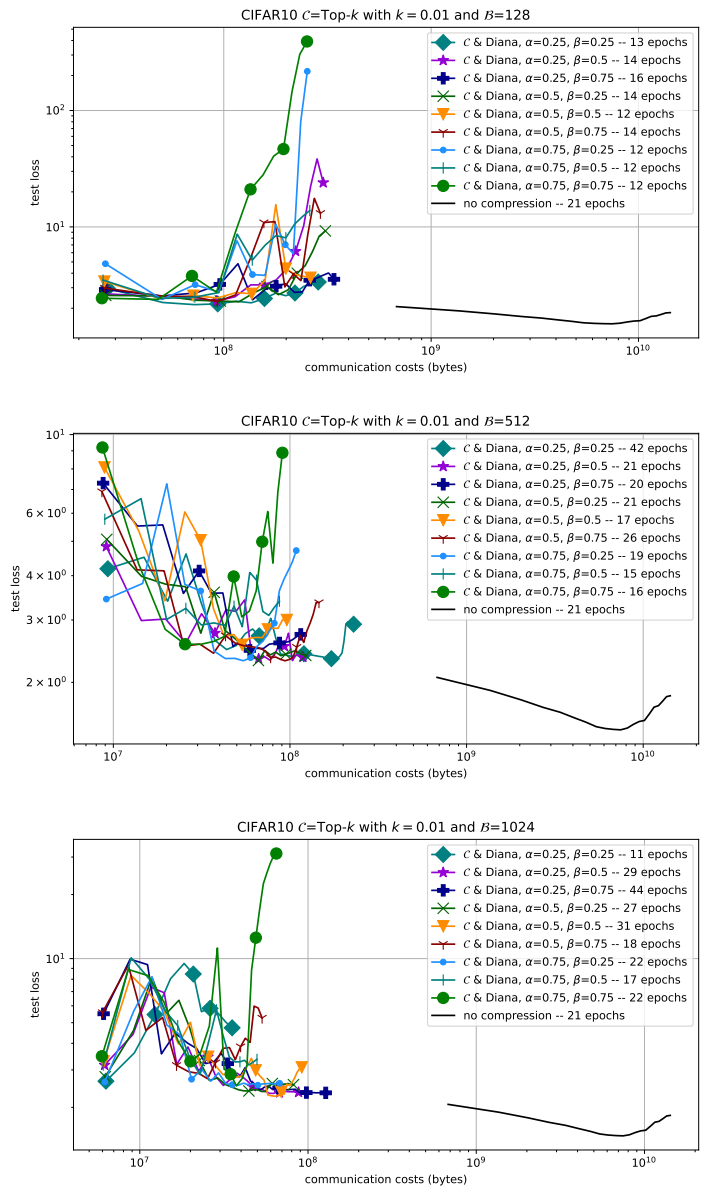


Figure 27: DIANA (Alg. 7) on MNIST under different batch sizes \mathcal{B} and hyperparameters, with $M = 3$ clients.



We first recall the following property satisfied μ -strongly convex and L -smooth functions.

Proposition B.2 ([Nes18, Theorem 2.1.12]). *For any μ -strongly convex and L -smooth function $f : \mathbb{R}^d \rightarrow \mathbb{R}$ (see Definitions 4.1 and 4.2), it holds, for any $x, y \in \mathbb{R}^d$,*

$$\langle \nabla f(x) - \nabla f(y), x - y \rangle \geq \frac{\mu L}{\mu + L} \|x - y\|^2 + \frac{1}{\mu + L} \|\nabla f(x) - \nabla f(y)\|^2.$$

Proof of Theorem 4.3. Let $t \in \mathbb{N}$ and $w^* = \arg \min f$ be the unique minimizer (by strong convexity). We have

$$\|w_{t+1} - w^*\|^2 = \|w_t - w^*\|^2 - 2\eta \left\langle w_t - w^*, \frac{1}{M} \sum_{i=1}^M D_{t+1}^i \right\rangle + \eta^2 \left\| \frac{1}{M} \sum_{i=1}^M D_{t+1}^i \right\|^2.$$

By Assumptions A1 and A3, we have

$$\mathbb{E}[\mathcal{C}((g_{t+1}^i)^\perp)] = \mathbb{E}[(g_{t+1}^i)^\perp].$$

Therefore, using Assumption A3 again, it follows that

$$\mathbb{E}[D_{t+1}^i] = \mathbb{E}[g_{t+1}^i] = \mathbb{E}[\nabla f_i(w_t)].$$

Conditioning on w_t , we obtain, using A3,

$$\mathbb{E}[\|w_{t+1} - w^*\|^2] = \mathbb{E}[\|w_t - w^*\|^2] - 2\eta \mathbb{E}[\langle w_t - w^*, \nabla f(w_t) \rangle] + \eta^2 \mathbb{E}\left[\left\| \frac{1}{M} \sum_{i=1}^M D_{t+1}^i \right\|^2\right].$$

Using Proposition B.2 (since $\nabla f(w^*) = 0$), it becomes

$$\begin{aligned} \mathbb{E}[\|w_{t+1} - w^*\|^2] &\leq \left(1 - 2\eta \frac{\mu L}{\mu + L}\right) \mathbb{E}[\|w_t - w^*\|^2] - 2\frac{\eta}{\mu + L} \mathbb{E}[\|\nabla f(w_t)\|] \\ &\quad + \eta^2 \mathbb{E}\left[\left\| \frac{1}{M} \sum_{i=1}^M D_{t+1}^i \right\|^2\right]. \end{aligned} \quad (6)$$

Using again A3 and the fact that $\mathbb{E}[D_{t+1}^i] = \mathbb{E}[\nabla f_i(w_t)]$, we have

$$\begin{aligned} \mathbb{E}\left[\left\| \frac{1}{M} \sum_{i=1}^M D_{t+1}^i \right\|^2\right] &= \mathbb{E}[\|\nabla f(w_t)\|^2] + \mathbb{E}\left[\left\| \frac{1}{M} \sum_{i=1}^M D_{t+1}^i - \nabla f_i(w_t) \right\|^2\right] \\ &= \mathbb{E}[\|\nabla f(w_t)\|^2] + \frac{1}{M^2} \sum_{i=1}^M \mathbb{E}[\|D_{t+1}^i - \nabla f_i(w_t)\|^2]. \end{aligned} \quad (7)$$

Using A1 and A3,

$$\begin{aligned} \mathbb{E}[\|D_{t+1}^i - \nabla f_i(w_t)\|^2] &= \mathbb{E}[\|D_{t+1}^i\|^2] - \mathbb{E}[\|\nabla f_i(w_t)\|^2] \\ &= \mathbb{E}[\|\alpha_{t+1}^i \bar{D}_k^i + \mathcal{C}((g_{t+1}^i)^\perp)\|^2] - \mathbb{E}[\|\nabla f_i(w_t)\|^2] \\ &= \mathbb{E}[\|\alpha_{t+1}^i \bar{D}_k^i\|^2] + \mathbb{E}[\|\mathcal{C}((g_{t+1}^i)^\perp)\|^2] - \mathbb{E}[\|\nabla f_i(w_t)\|^2] \\ &\leq \mathbb{E}[\|\alpha_{t+1}^i \bar{D}_k^i\|^2] + \beta \mathbb{E}[\|(g_{t+1}^i)^\perp\|^2] - \mathbb{E}[\|\nabla f_i(w_t)\|^2] \\ &\leq \beta \mathbb{E}[\|\alpha_{t+1}^i \bar{D}_k^i\|^2] + \beta \mathbb{E}[\|(g_{t+1}^i)^\perp\|^2] - \mathbb{E}[\|\nabla f_i(w_t)\|^2] \quad (\beta \geq 1) \\ &= \beta \mathbb{E}[\|g_{t+1}^i\|^2] - \mathbb{E}[\|\nabla f_i(w_t)\|^2] \leq (\beta - 1) \mathbb{E}[\|\nabla f_i(w_t)\|^2] + \beta \sigma^2. \end{aligned}$$

Going back to (7), we have

$$\mathbb{E} \left[\left\| \frac{1}{M} \sum_{i=1}^M D_{t+1}^i \right\|^2 \right] \leq \mathbb{E}[\|\nabla f(w_t)\|^2] + \frac{\beta-1}{M^2} \sum_{i=1}^M \mathbb{E}[\|\nabla f_i(w_t)\|^2] + \frac{\beta\sigma^2}{M}.$$

By A2,

$$\mathbb{E} \left[\left\| \frac{1}{M} \sum_{i=1}^M D_{t+1}^i \right\|^2 \right] \leq (1 + b \frac{\beta-1}{M}) \mathbb{E}[\|\nabla f(w_t)\|^2] + a \frac{\beta-1}{M} + \frac{\beta\sigma^2}{M}.$$

Going back to (6) we have

$$\begin{aligned} \mathbb{E}[\|w_{t+1} - w^*\|^2] &\leq \left(1 - 2\eta \frac{\mu L}{\mu + L}\right) \mathbb{E}[\|w_t - w^*\|^2] \\ &\quad + \eta \left(\eta \left(1 + b \frac{\beta-1}{M}\right) - \frac{2}{\mu + L} \right) \mathbb{E}[\|\nabla f(w_t)\|^2] \\ &\quad + \eta^2 a \frac{\beta-1}{M} + \eta^2 \frac{\beta\sigma^2}{M}. \end{aligned}$$

Since $\eta \left(1 + b \frac{\beta-1}{M}\right) \leq \frac{2}{\mu+L}$, we obtain

$$\mathbb{E}[\|w_{t+1} - w^*\|^2] \leq \left(1 - 2\eta \frac{\mu L}{\mu + L}\right) \mathbb{E}[\|w_t - w^*\|^2] + \eta^2 a \frac{\beta-1}{M} + \eta^2 \frac{\beta\sigma^2}{M}.$$

Hence, noticing⁵ that $\eta < \frac{\mu+L}{2\mu L}$, we have

$$\mathbb{E}[\|w_t - w^*\|^2] \leq \left(1 - 2\eta \frac{\mu L}{\mu + L}\right)^t \mathbb{E}[\|w_0 - w^*\|^2] + \eta \frac{\mu + L}{2\mu LM} (a(\beta-1) + \beta\sigma^2).$$

The proof is complete. □

C Proof of Theorem 4.4

This section is devoted to the proof of Theorem 4.4, restated below:

Theorem C.1. Assume A1 and that each f_i is convex and L -smooth (see Definition 4.2). Then, for any learning rate $0 < \eta < \frac{1}{\beta L}$, the sequence $(w_t)_{t \geq 0}$ generated by Algorithm 1 with deterministic gradients satisfies, for any $T \geq 1$,

$$\mathbb{E} \left[f \left(\frac{1}{T} \sum_{t=0}^{T-1} w_t \right) \right] - f^* \leq \frac{\|w_0 - w^*\|^2}{\eta T} + \frac{1}{LM \left(\frac{1}{\beta L \eta} - 1 \right)} \sum_{i=1}^M \|\nabla f_i(w^*)\|^2,$$

where $w^* \in \arg \min_{\mathbb{R}^d} f$ is any optimal point.

⁵Note that our assumption $\eta \leq (1 + b \frac{\beta-1}{M})^{-1} \frac{2}{\mu+L}$ implies $\eta < \frac{\mu+L}{2\mu L}$ as soon as $b(\beta-1) > 0$ or $L > \mu$.

We consider deterministic gradient descent for Algorithm 1, *i.e.*, we consider the following scheme: for $t \geq 0$ and $i \in [M]$,

$$\begin{cases} D_0^i = 0, & w_0 \in \mathbb{R}^d \\ \bar{D}_t^i = \frac{1}{K} \sum_{k=0}^{K-1} D_k^i & \text{(see the comment line 6 of Alg. 1),} \\ g_{t+1}^i = \nabla f_i(w_t) = \alpha_{t+1}^i \bar{D}_t^i + (g_{t+1}^i)^\perp, & \text{with } \bar{D}_t^i \perp (g_{t+1}^i)^\perp, \\ D_{t+1}^i = \alpha_{t+1}^i \bar{D}_t^i + \mathcal{C}((g_{t+1}^i)^\perp) \\ w_{t+1} = w_t - \frac{\eta}{M} \sum_{i=1}^M D_{t+1}^i, & t \geq 0. \end{cases} \quad (8)$$

To prove Theorem 4.4, we first recall the following result.

Proposition C.2 ([Nes18, Theorem 2.1.5]). *A function $f : \mathbb{R}^d \rightarrow \mathbb{R}$ is a convex, differentiable and L -smooth (see Definition 4.2) if and only if it satisfies:*

$$0 \leq f(y) - f(x) - \langle \nabla f(x), y - x \rangle \leq \frac{L}{2} \|x - y\|^2, \quad \forall x, y \in \mathbb{R}^d, \quad (9)$$

which is equivalent to

$$\|\nabla f(x) - \nabla f(y)\|^2 \leq L \langle \nabla f(x) - \nabla f(y), x - y \rangle, \quad \forall x, y \in \mathbb{R}^d. \quad (10)$$

Our proof is inspired by the proof of [KFJ18, Theorem 5.3].

Proof of Theorem 4.4. Let $t \in \mathbb{N}$. Denote by w^* any optimal point (*i.e.*, $f(w^*) = \frac{1}{M} \sum_{i=1}^M f_i(w^*) = f^* = \min_{\mathbb{R}^d} f$). We have, by (8) and convexity,

$$\begin{aligned} \|w_{t+1} - w^*\|^2 &= \|w_t - w^*\|^2 - 2\frac{\eta}{M} \langle w_t - w^*, \sum_{i=1}^M D_{t+1}^i \rangle + \frac{\eta^2}{M^2} \left\| \sum_{i=1}^M D_{t+1}^i \right\|^2 \\ &\leq \|w_t - w^*\|^2 - 2\frac{\eta}{M} \langle w_t - w^*, \sum_{i=1}^M D_{t+1}^i \rangle + \frac{\eta^2}{M} \sum_{i=1}^M \|D_{t+1}^i\|^2. \end{aligned}$$

By Assumption A1,

$$\mathbb{E}[\|w_{t+1} - w^*\|^2] \leq \mathbb{E}[\|w_t - w^*\|^2] - 2\frac{\eta}{M} \mathbb{E} \left[\langle w_t - w^*, \sum_{i=1}^M \nabla f_i(w_t) \rangle \right] + \frac{\eta^2}{M} \sum_{i=1}^M \mathbb{E}[\|D_{t+1}^i\|^2]. \quad (11)$$

Let $i \in [M]$. By orthogonality and A1, it holds $\mathbb{E}[\|D_{t+1}^i\|^2] = \mathbb{E}[\|\alpha_{t+1}^i \bar{D}_t^i\|^2 + \|\mathcal{C}((g_{t+1}^i)^\perp)\|^2]$ and $\mathbb{E}[\|\mathcal{C}((g_{t+1}^i)^\perp)\|^2] \leq \beta \mathbb{E}[\|(g_{t+1}^i)^\perp\|^2]$. Hence, since $\beta \geq 1$,

$$\mathbb{E}[\|D_{t+1}^i\|^2] \leq \beta \mathbb{E}[\|\alpha_{t+1}^i \bar{D}_t^i\|^2 + \|(g_{t+1}^i)^\perp\|^2] = \beta \mathbb{E}[\|\nabla f_i(w_t)\|^2]. \quad (12)$$

Using that $\|x\|^2 = \|x - y\|^2 + \|y\|^2 + 2\langle x - y, y \rangle$ for any $x, y \in \mathbb{R}^d$, together with Young's inequality, it holds, for any $\epsilon > 0$,

$$\begin{aligned} \|\nabla f_i(w_t)\|^2 &= \|\nabla f_i(w_t) - \nabla f_i(w^*)\|^2 + \|\nabla f_i(w^*)\|^2 + 2\langle \nabla f_i(w_t) - \nabla f_i(w^*), \nabla f_i(w^*) \rangle \\ &\leq \|\nabla f_i(w_t) - \nabla f_i(w^*)\|^2 + \|\nabla f_i(w^*)\|^2 + \epsilon \|\nabla f_i(w_t) - \nabla f_i(w^*)\|^2 + \frac{1}{\epsilon} \|\nabla f_i(w^*)\|^2 \\ &= (1 + \epsilon) \|\nabla f_i(w_t) - \nabla f_i(w^*)\|^2 + \left(1 + \frac{1}{\epsilon}\right) \|\nabla f_i(w^*)\|^2 \\ &\stackrel{(10)}{\leq} (1 + \epsilon) L \langle \nabla f_i(w_t) - \nabla f_i(w^*), w_t - w^* \rangle + \left(1 + \frac{1}{\epsilon}\right) \|\nabla f_i(w^*)\|^2. \end{aligned}$$

Hence, we have

$$\mathbb{E}[\|D_{t+1}^i\|^2] \leq \beta(1 + \epsilon)L\mathbb{E}[\langle \nabla f_i(w_t) - \nabla f_i(w^*), w_t - w^* \rangle] + \beta\left(1 + \frac{1}{\epsilon}\right)\|\nabla f_i(w^*)\|^2. \quad (13)$$

Plugging (13) into (11), we obtain, using also that $\nabla f(w^*) = \frac{1}{M} \sum_{i=1}^M \nabla f_i(w^*) = 0$,

$$\begin{aligned} \mathbb{E}[\|w_{t+1} - w^*\|^2] &\leq \mathbb{E}[\|w_t - w^*\|^2] - 2\eta\mathbb{E}\left[\left\langle w_t - w^*, \frac{1}{M} \sum_{i=1}^M \nabla f_i(w_t) - f(w^*) \right\rangle\right] \\ &\quad + \beta(1 + \epsilon)L\eta^2\mathbb{E}\left[\left\langle \frac{1}{M} \sum_{i=1}^M \nabla f_i(w_t) - \nabla f_i(w^*), w_t - w^* \right\rangle\right] \\ &\quad + \left(1 + \frac{1}{\epsilon}\right)\frac{\beta\eta^2}{M} \sum_{i=1}^M \|\nabla f_i(w^*)\|^2 \\ &= \mathbb{E}[\|w_t - w^*\|^2] + \eta(\beta(1 + \epsilon)L\eta - 2)\mathbb{E}[\langle \nabla f(w_t) - \nabla f(w^*), w_t - w^* \rangle] \\ &\quad + \left(1 + \frac{1}{\epsilon}\right)\frac{\beta\eta^2}{M} \sum_{i=1}^M \|\nabla f_i(w^*)\|^2. \end{aligned}$$

Since $\eta < \frac{1}{\beta L}$, there exists $\epsilon > 0$ such that $\eta = \frac{1}{\beta(1+\epsilon)L}$. With this value, the last inequality reads

$$\begin{aligned} \mathbb{E}[\|w_{t+1} - w^*\|^2] &\leq \mathbb{E}[\|w_t - w^*\|^2] - \eta\mathbb{E}[\langle \nabla f(w_t) - \nabla f(w^*), w_t - w^* \rangle] \\ &\quad + \frac{1}{\beta(1 + \epsilon)L^2\epsilon M} \sum_{i=1}^M \|\nabla f_i(w^*)\|^2. \end{aligned} \quad (14)$$

By convexity of f , $-\eta\mathbb{E}[\langle \nabla f(w_t) - \nabla f(w^*), w_t - w^* \rangle] \leq -\eta\mathbb{E}[f(w_t) - f^*]$. Hence, from (14), we obtain

$$\eta\mathbb{E}[f(w_t) - f^*] \leq \mathbb{E}[\|w_t - w^*\|^2] - \mathbb{E}[\|w_{t+1} - w^*\|^2] + \frac{1}{\beta(1 + \epsilon)L^2\epsilon M} \sum_{i=1}^M \|\nabla f_i(w^*)\|^2.$$

Introducing, for $T \in \mathbb{N}^*$, $\bar{w}_T = \frac{1}{T} \sum_{t=0}^{T-1} w_t$, we obtain, using the last inequality together with the convexity inequality $f(\bar{w}_T) \leq \frac{1}{T} \sum_{t=0}^{T-1} f(w_t)$,

$$\begin{aligned} \mathbb{E}[f(\bar{w}_T) - f^*] &\leq \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[f(w_t) - f^*] \\ &\leq \frac{1}{\eta T} \sum_{t=0}^{T-1} \left[\mathbb{E}[\|w_t - w^*\|^2] - \mathbb{E}[\|w_{t+1} - w^*\|^2] + \frac{1}{\beta(1 + \epsilon)L^2\epsilon M} \sum_{i=1}^M \|\nabla f_i(w^*)\|^2 \right] \\ &\leq \frac{\mathbb{E}[\|w_0 - w^*\|^2]}{\eta T} + \frac{1}{\eta\beta(1 + \epsilon)L^2\epsilon M} \sum_{i=1}^M \|\nabla f_i(w^*)\|^2 \\ &= \frac{\mathbb{E}[\|w_0 - w^*\|^2]}{\eta T} + \frac{1}{L\epsilon M} \sum_{i=1}^M \|\nabla f_i(w^*)\|^2, \end{aligned}$$

where we used the non-negativity of the norm and the fact that $\eta\beta(1 + \epsilon)L = 1$. The proof is complete. \square

References

- [AGL⁺17] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic. Qsgd: Communication-efficient sgd via gradient quantization and encoding. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.
- [AH17] A. F. Aji and K. Heafield. Sparse communication for distributed gradient descent. In M. Palmer, R. Hwa, and S. Riedel, editors, *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 440–445, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.
- [AHJ⁺18] D. Alistarh, T. Hoefler, M. Johansson, N. Konstantinov, S. Khirirat, and C. Renggli. The convergence of sparsified gradient methods. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.
- [BCN18] L. Bottou, F. E. Curtis, and J. Nocedal. Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311, 2018.
- [BHRS23] A. Beznosikov, S. Horváth, P. Richtárik, and M. Safaryan. On biased compression for distributed learning. *Journal of Machine Learning Research*, 24(276):1–50, 2023.
- [CR22] L. Condat and P. Richtarik. Murana: A generic framework for stochastic variance-reduced optimization. In Bin Dong, Qianxiao Li, Lei Wang, and Zhi-Qin John Xu, editors, *Proceedings of Mathematical and Scientific Machine Learning*, volume 190 of *Proceedings of Machine Learning Research*, pages 81–96. PMLR, 15–17 Aug 2022.
- [DBA⁺20] A. Dutta, E. H. Bergou, A. M. Abdelmoniem, C. Ho, A. N. Sahu, M. Canini, and P. Kalnis. On the discrepancy between the theoretical analysis and practical implementations of compressed communication for distributed deep learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):3817–3824, Apr. 2020.
- [Det16] T. Dettmers. 8-bit approximations for parallelism in deep learning. In *International Conference on Learning Representations (ICLR)*, 2016.
- [DMJVE16] N. Dryden, T. Moon, S. A. Jacobs, and B. Van Essen. Communication quantization for data-parallel training of deep neural networks. In *2016 2nd Workshop on Machine Learning in HPC Environments (MLHPC)*, pages 1–8, 2016.
- [FSG⁺21] I. Fatkhullin, I. Sokolov, E. Gorbunov, Z. Li, and P. Richtárik. Ef21 with bells & whistles: Practical algorithmic extensions of modern error feedback. *arXiv preprint arXiv:2110.03294*, 2021.
- [FTR23] I. Fatkhullin, A. Tyurin, and P. Richtarik. Momentum provably improves error feedback! In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and

- S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 76444–76495. Curran Associates, Inc., 2023.
- [GHR20] E. Gorbunov, F. Hanzely, and P. Richtarik. A unified theory of sgd: Variance reduction, sampling, quantization and coordinate descent. In Silvia Chiappa and Roberto Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 680–690. PMLR, 26–28 Aug 2020.
- [GTR23] K. Gruntkowska, A. Tyurin, and P. Richtárik. EF21-p and friends: Improved theoretical communication complexity for distributed optimization with bidirectional compression. In A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 11761–11807. PMLR, 23–29 Jul 2023.
- [HHY23] Y. He, X. Huang, and K. Yuan. Unbiased compression saves communication in distributed optimization: When and how much? In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 47991–48020. Curran Associates, Inc., 2023.
- [HKM⁺23] S. Horváth, D. Kovalev, K. Mishchenko, P. Richtárik, and S. Stich. Stochastic distributed learning with gradient quantization and double-variance reduction. *Optimization Methods and Software*, 38(1):91–106, 2023.
- [HZRS16] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.
- [Ide] Y. Idelbayev. Proper ResNet implementation for CIFAR10/CIFAR100 in PyTorch. https://github.com/akamaster/pytorch_resnet_cifar10. Accessed: 20xx-xx-xx.
- [KFJ18] S. Khirirat, H. R. Feyzmahdavian, and M. Johansson. Distributed learning with compressed gradients. *arXiv preprint arXiv:1806.06573*, 2018.
- [KKM⁺20] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh. SCAFFOLD: Stochastic controlled averaging for federated learning. In H. D. III and A. Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5132–5143. PMLR, 13–18 Jul 2020.
- [KMA⁺21] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, R. G. L. D’Oliveira, H. Eichner, S. El Rouayheb, D. Evans, J. Gardner, Z. Garrett, A. Gascón, B. Ghazi, P. B. Gibbons, M. Gruteser, Z. Harchaoui, C. He, L. He, Z. Huo, B. Hutchinson, J. Hsu, M. Jaggi, T. Javidi, G. Joshi, M. Khodak, J. Konecný, A. Korolova, F. Koushanfar, S. Koyejo, T. Lepoint, Y. Liu, P. Mittal, M. Mohri, R. Nock, A. Özgür, R. Pagh, H. Qi, D. Ramage, R. Raskar, M. Raykova, D. Song, W. Song, S. U. Stich, Z. Sun, A. T. Suresh, F. Tramèr, P. Vepakomma, J. Wang, L. Xiong, Z. Xu, Q. Yang, F. X.

- Yu, H. Yu, and S. Zhao. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021.
- [KRSJ19] S. P. Karimireddy, Q. Rebjock, S. Stich, and M. Jaggi. Error feedback fixes SignSGD and other gradient compression schemes. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3252–3261. PMLR, 09–15 Jun 2019.
- [L⁺15] Y. LeCun et al. Lenet-5, convolutional neural networks. URL: <http://yann.lecun.com/exdb/lenet>, 20(5):14, 2015.
- [LKQR20] Z. Li, D. Kovalev, X. Qian, and P. Richtarik. Acceleration for compressed gradient descent in distributed and federated optimization. In H. D. III and A. Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5895–5904. PMLR, 13–18 Jul 2020.
- [MGI⁺22] M. Makarenko, E. Gasanov, R. Islamov, A. Sadiev, and P. Richtárik. Adaptive compression for communication-efficient distributed training. *arXiv preprint arXiv:2211.00188*, 2022.
- [MGTa24] K. Mishchenko, E. Gorbunov, M. Takáč, and P. Richtárik and. Distributed learning with compressed gradient differences*. *Optimization Methods and Software*, 0(0):1–16, 2024.
- [MMR⁺17] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. In A. Singh and J. Zhu, editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 1273–1282. PMLR, 20–22 Apr 2017.
- [Nes18] Y. Nesterov. *Lectures on convex optimization*, volume 137. Springer, 2018.
- [PD24] C. Philippenko and A. Dieuleveut. Compressed and distributed least-squares regression: convergence rates with applications to federated learning. *Journal of Machine Learning Research*, 25(288):1–80, 2024.
- [RSF21] P. Richtarik, I. Sokolov, and I. Fatkhullin. Ef21: A new, simpler, theoretically better, and practically faster error feedback. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 4384–4396. Curran Associates, Inc., 2021.
- [SCJ18] S. U. Stich, J-B. Cordonnier, and M. Jaggi. Sparsified sgd with memory. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.
- [SFD⁺14] F. Seide, H. Fu, J. Droppo, G. Li, and D. Yu. 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech dnns. In *Interspeech*, volume 2014, pages 1058–1062. Singapore, 2014.

- [SSR21] M. Safaryan, E. Shulgin, and P. Richtárik. Uncertainty principle for communication compression in distributed and federated learning and the search for an optimal compressor. *Information and Inference: A Journal of the IMA*, 11(2):557–580, 04 2021.
- [TYL⁺19] H. Tang, C. Yu, X. Lian, T. Zhang, and J. Liu. DoubleSqueeze: Parallel stochastic gradient descent with double-pass error-compensated compression. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6155–6165. PMLR, 09–15 Jun 2019.
- [XHA⁺21] H. Xu, C-Y. Ho, A. M. Abdelmoniem, A. Dutta, E. H. Bergou, K. Karatsenidis, M. Canini, and P. Kalnis. Grace: A compressed communication framework for distributed machine learning. In *2021 IEEE 41st international conference on distributed computing systems (ICDCS)*, pages 561–572. IEEE, 2021.
- [ZLR21] H. Zhao, Z. Li, and P. Richtárik. Fedpage: A fast local stochastic gradient method for communication-efficient federated learning. *arXiv preprint arXiv:2108.04755*, 2021.