INDIAN STATISTICAL INSTITUTE, KOLKATA
Cryptology and Security

Dynamic Sparsification in Secure Gradient Aggregation for Federated Learning

by

Bikash Samanta

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Master of Technology

July 2025

Declaration

I hereby declare that this dissertation entitled *Dynamic Sparsification in Secure Gradient Aggregation for Federated Learning*, submitted to Indian Statistical Institute, Kolkata in partial fulfillment of the requirements for the award of the degree Master of Technology in Cryptology and Security, is a record of independent research work carried out by me under the supervision of:

- Dr. Pradip Sasmal, Professor, Indian Institute of Technology Jodhpur.
- Dr. Anisur Rahman Molla, Faculty Member, Indian Statistical Institute, Kolkata.

This work is an extension of previously published research, and appropriate references have been provided wherever applicable. The results presented are my own contributions and have not been submitted elsewhere for the award of any other degree or diploma.

All sources of information have been duly acknowledged.

Bikash Samanta
July 2025

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my supervisors, Dr. Pradip Sasmal and Dr. Anisur Rahman Molla, for their invaluable guidance, continuous support, and encouragement throughout the course of this research. Their deep insights, constructive feedback, and generous availability of time have been instrumental in shaping this dissertation.

I am grateful to the faculty and staff of the Cryptology and Security at Indian Statistical Institute, Kolkata for providing a stimulating academic environment and the resources necessary to carry out my research.

I also wish to thank my peers and friends at the Indian Statistical Institute, Kolkata, for their camaraderie, helpful discussions, and moral support during the course of this program.

A special note of thanks to my family for their unwavering encouragement, patience, and support throughout my academic journey. Their belief in me has been my source of strength.

Finally, I acknowledge all the researchers whose work has inspired and informed my own. This dissertation would not have been possible without the foundation laid by prior research in the field.

Bikash Samanta

Abstract

Secure aggregation is a critical component of privacy-preserving federated learning. However, existing fixed-sparsity approaches often incur unnecessary communication overhead. We present **DynamicSecAgg**, a novel framework that introduces dynamic sparsity while preserving coordinate-level privacy. Our method achieves significant improvements in communication efficiency while maintaining — and in some cases improving — model accuracy across both IID and non-IID user distributions. The framework maintains information-theoretic privacy guarantees via adaptive gradient thresholding and polynomial-based aggregation, proving particularly effective under heterogeneous data settings. These results establish dynamic sparsity as a key optimization for efficient and privacy-preserving federated learning.

Contents

Chapter 1

Introduction

Privacy-preserving machine learning has become a crucial requirement in distributed settings, where multiple users collaborate to train a shared model without exposing their raw data. Homomorphic encryption (HE) has emerged as a promising cryptographic solution in this domain, as it allows computations to be performed directly on encrypted data. In particular, partially homomorphic encryption (PHE) schemes such as Paillier encryption enable additive operations over ciphertexts, allowing encrypted gradients to be aggregated without decryption. While HE offers strong theoretical privacy guarantees, its practical application in federated learning is severely limited by large ciphertext sizes, high computation overhead, and increased latency, especially in resource-constrained environments.

As a more efficient alternative, secure aggregation (SA) has gained traction. SA is a multi-party computation technique where users mask their local gradients using random values and send only the masked data to a central server. These masks cancel out upon aggregation, allowing the server to recover the sum of the gradients without learning anything about individual contributions. SA offers a compelling balance between privacy and computational efficiency. Furthermore, it presents an opportunity to shift communication-intensive tasks to an offline phase—before training begins—thus reducing the computational burden during the online phase.

To address the communication bottleneck in federated learning, sparsification techniques have been widely adopted. In sparsified learning, each user transmits only a small subset of gradient coordinates instead of the full gradient vector. This significantly reduces communication overhead, but introduces a new form of vulnerability: the exposure of coordinate information. Over multiple training rounds, this coordinate leakage can be exploited by adversaries to reconstruct sensitive user data, even if the gradients are securely aggregated.

TinySecAgg was recently proposed to overcome this challenge. It extends secure aggregation by not only masking the gradient values but also hiding their corresponding coordinates. This coordinate-hiding property significantly strengthens privacy guarantees under sparsified learning. TinySecAgg also adopts an offline-online computation model to reduce the real-time workload on user devices by performing the most demanding steps in advance.

While TinySecAgg offers a solid foundation, it assumes a fixed sparsity level

throughout training. In this work, we propose an enhanced framework that introduces dynamic sparsity into the TinySecAgg protocol. Our method adapts the sparsification level over time based on the training dynamics, allowing the system to balance communication cost, and model performance more effectively. By integrating dynamic sparsity with coordinate-hiding secure aggregation, and leveraging offline computation, our framework achieves improved efficiency and learning quality in practical federated learning deployments.

Chapter 2

Related Works

Gradient Inversion Attacks

Federated learning offers privacy benefits by keeping raw data on user devices, but recent studies have shown that gradients themselves can leak sensitive information. In gradient inversion attacks, adversaries aim to reconstruct local input data from shared gradients. Early methods optimize synthetic inputs to match the observed gradients, minimizing the Euclidean distance to the true ones. Other approaches exploit relationships between gradient signs and labels or use cosine similarity for more accurate recovery. Advanced techniques also consider batch-level attacks and exploit dropout patterns or model modifications to amplify leakage.

These attacks expose the limitations of naive gradient sharing, even under aggregation schemes. It becomes critical to deploy additional mechanisms that obscure or protect the content and structure of the gradients transmitted by clients.

Secure Aggregation Protocols

Secure aggregation (SA) protocols were developed to prevent the server from accessing individual gradients. By leveraging secure multi-party computation, each user masks their local gradient before sending it to the server. The server can only recover the aggregated result, not the individual contributions. Classic SA protocols such as SecAgg and its variants use random masking and secret sharing to achieve these guarantees, typically assuming an honest-but-curious adversary model.

Although SA can be extended to support gradient compression methods like sparsification, it still allows the server to observe the coordinates selected by each user, which can be exploited in multi-round attacks to reconstruct the original data.

Sparsification and Communication Efficiency

Due to bandwidth constraints in federated learning, communication efficiency is a major concern. Gradient sparsification addresses this issue by letting users send only a subset of their gradient entries—typically the largest ones (top-K) or a random subset (rand-K). This significantly reduces communication overhead. A complementary

approach is gradient quantization, which encodes gradient values using fewer bits.

However, sparsification introduces privacy risks. If the server knows which coordinates are selected by each user, it can accumulate this information over multiple rounds to infer user-specific data. This has led to increased focus on coordinate-hiding mechanisms.

Coordinate-Hiding and Private Information Retrieval

Recent works explore coordinate-hiding strategies to counteract the risks introduced by sparsification. These include techniques inspired by private information retrieval (PIR), where users interact with multiple non-colluding servers to hide the identity of accessed or updated coordinates. Though powerful, PIR-based methods often assume a multi-server setup and may not be applicable to standard federated learning systems with a single server.

TinySecAgg addresses this limitation by enabling coordinate-hiding in a single-server federated learning setup. It extends traditional secure aggregation protocols by masking both gradient values and their coordinates. This is achieved via structured encoding techniques in an offline phase, followed by efficient aggregation in an online phase.

Positioning of This Work

While TinySecAgg marks a major step forward in secure, sparsified aggregation, it operates under a fixed sparsity rate throughout the training process. In contrast, our proposed framework enhances TinySecAgg by introducing dynamic sparsity, allowing the sparsification level to adapt over time based on training progress and performance metrics. This improves learning flexibility and communication efficiency while preserving privacy through coordinate-hiding.

Chapter 3

Problem Formulation

We consider a centralized federated learning architecture consisting of N users, coordinated by a central server. Each user $i \in [N]$ holds a local dataset $\mathcal{D}_i$ containing $|\mathcal{D}_i|$ samples. The goal is to collaboratively train a global model $w \in \mathbb{R}^d$ by minimizing the global loss:

$$\min_w F(w) \triangleq \sum_{i=1}^{N} F_i(w), \tag{3.1}$$

where $F_i(w)$ is the local loss function associated with user i.

Training proceeds in rounds. At the beginning of round t, the server sends the current global model w^t to all users. Each user initializes their local model $w_i^t \leftarrow w^t$, and performs E local training steps using gradient descent:

$$w_i^t \leftarrow w_i^t - \eta \nabla F_i(w_i^t), \tag{3.2}$$

where η is the learning rate and $\nabla F_i(w_i^t)$ is the gradient evaluated on $\mathcal{D}_i$.

After local updates, user i computes the cumulative local update (gradient difference):

$$\Delta_i^t \triangleq w_i^t - w^t. \tag{3.3}$$

The server aggregates these updates from the set of participating users $U(t)$ (some may drop out), and updates the global model:

$$w^{t+1} = w^t - \frac{1}{|U(t)|} \sum_{i \in U(t)} \Delta_i^t. \tag{3.4}$$

To reduce communication overhead, gradient sparsification is employed. Instead of sending the full update Δ_i^t, each user selects only $K \ll d$ parameters and sends a sparsified update. The sparsified gradient is defined as:

$$x_i^t \triangleq b_i^t \odot \tilde{\Delta}_i^t, \tag{3.5}$$

where $b_i^t \in \{0,1\}^d$ is a binary mask indicating the selected coordinates (with $\|b_i^t\|_1 = K$), $\odot$ is the Hadamard product, and

$$\tilde{\Delta}_i^t \triangleq \Delta_i^t + e_i^{t-1} \tag{3.6}$$

includes the accumulated error e_i^{t-1} from previous rounds. The error accumulation is updated as:

$$e_i^t \triangleq \Delta_i^t + e_i^{t-1} - x_i^t = \tilde{\Delta}_i^t - x_i^t. \tag{3.7}$$

Each user sends the K non-zero entries of x_i^t and the corresponding coordinate indices to the server. The server then aggregates the received sparsified updates as:

$$w^{t+1} = w^t - \frac{1}{|U(t)|} \sum_{i \in U(t)} x_i^t. \tag{3.8}$$

Two popular sparsification schemes are:

- **Rand-K sparsification:** User i selects K random coordinates uniformly without replacement. The binary mask b_i^t is generated randomly and can be precomputed offline.
- **Top-K sparsification:** User i selects the K coordinates with the largest magnitudes in $\tilde{\Delta}_i^t$.

While Top-K offers faster convergence, it requires real-time computation and online transmission of coordinate indices. In contrast, Rand-K is more communication-efficient and can leverage offline generation of masks.

Threat Model

We adopt an honest-but-curious adversary model, where the server and up to $T < N$ users may behave adversarially. The adversarial server follows the protocol honestly in terms of computation and message passing but seeks to infer private information from the honest users. Similarly, up to T clients may collude with the server to gain additional information. Let $\mathcal{H}$ denote the set of honest users and $\mathcal{T} = [N] \setminus \mathcal{H}$ denote the set of adversarial users.

Secure Aggregation Objective

The goal of secure aggregation is to ensure that the server learns only the sum of the masked user updates, without revealing any individual updates or coordinate choices.

Let x_i^t be the masked and sparsified update sent by user i in round t, and let $U(t) \subseteq [N]$ be the set of users who successfully participate in that round. The server should be able to compute only the aggregate:

$$x_{\text{agg}}^t \triangleq \sum_{i \in U(t)} x_i^t, \tag{3.9}$$

while gaining no additional information about the individual x_i^t of honest users.

This privacy guarantee is captured through the following information-theoretic condition:

$$I\left(\{x_i^t\}_{i \in \mathcal{H}}; M_T^t \mid x_{\text{agg}}^t, \{x_i^t\}_{i \in \mathcal{T}}, R_T^t\right) = 0 \tag{3.10}$$

where:

- $I(\cdot;\cdot \mid \cdot)$ is the conditional mutual information.
- $\{x_i^t\}_{i \in \mathcal{H}}$ is the set of updates from honest users.
- M_T^t represents all messages observed by the adversarial server and clients during round t.
- $\{x_i^t\}_{i \in \mathcal{T}}$ are the updates from the colluding (adversarial) users.
- R_T^t is the randomness generated by the adversaries and server.

This condition guarantees **information-theoretic privacy**, meaning that the adversary learns nothing about the honest users' individual updates beyond what is already revealed by the aggregate and its own knowledge.

Correctness Guarantee

In addition to privacy, the protocol must ensure correctness—that the server can reconstruct the correct aggregate update from the received messages. This is formalized as:

$$H(x_{\text{agg}}^t \mid M_{U(t)}^t) = 0 \tag{3.11}$$

where:

- $H(\cdot \mid \cdot)$ is the conditional entropy.
- x_{agg}^t is the true aggregate of updates from participating users.
- $M_{U(t)}^t$ is the set of messages received from all users in $U(t)$ during round t.

This condition ensures that the aggregation result is **deterministically computable** from the observed messages, i.e., the protocol is correct and complete when the required number of users participate.

Chapter 4

Reconstruction from Compressed Gradients

Gradient sparsification can significantly reduce communication overhead in federated learning, especially when combined with secure aggregation. However, it introduces a unique vulnerability: the server may be able to recover individual gradients using the coordinate information shared during sparsification, even if only the sum of masked gradients is revealed.

In conventional secure aggregation protocols, each user applies additive random masks to hide their selected gradient entries. These masks are constructed such that they cancel out during aggregation, allowing the server to compute the sum of user updates:

$$x_{\text{agg}}^t = \sum_{i \in U(t)} x_i^t, \tag{4.1}$$

without revealing the individual values x_i^t. However, if the coordinates of the selected entries are visible to the server, the privacy guarantee can be compromised.

We consider a **frozen-model setup**, where the global model is fixed, resulting in constant local gradients over time, i.e., $\Delta_i^t = \Delta_i$. This scenario can also arise naturally when the model has nearly converged. Under this assumption, the gradient accumulated with error feedback can be expressed at each coordinate ℓ as:

$$\tilde{\Delta}_i^t(\ell) = (t - \tau_i^t(\ell))\Delta_i(\ell), \tag{4.2}$$

where $\tau_i^t(\ell) < t$ is the most recent round prior to t in which user i sent coordinate ℓ to the server.

Thus, the transmitted sparsified gradient is:

$$x_i^t(\ell) = b_i^t(\ell) \cdot \tilde{\Delta}_i^t(\ell), \tag{4.3}$$

where $b_i^t(\ell)$ is the binary mask indicating whether coordinate ℓ was selected by user i at round t.

The server learns the aggregate at each coordinate ℓ as:

$$x_{\text{agg}}^t(\ell) = \sum_{i \in U(t)} x_i^t(\ell) = \sum_{i \in U(t)} b_i^t(\ell)(t - \tau_i^t(\ell))\Delta_i(\ell). \tag{4.4}$$

By collecting the aggregated values over J rounds, the server obtains the following linear system:

$$A \begin{bmatrix} \Delta_1(\ell) \\ \vdots \\ \Delta_N(\ell) \end{bmatrix} + n = \begin{bmatrix} x_{\text{agg}}^1(\ell) \\ \vdots \\ x_{\text{agg}}^J(\ell) \end{bmatrix}, \quad \forall \ell \in [d], \tag{4.5}$$

where $A \in \mathbb{R}^{J \times N}$ is defined as:

$$A = \begin{bmatrix} b_1^1(\ell)(1 - \tau_1^1(\ell)) & \cdots & b_N^1(\ell)(1 - \tau_N^1(\ell)) \\ \vdots & \ddots & \vdots \\ b_1^J(\ell)(J - \tau_1^J(\ell)) & \cdots & b_N^J(\ell)(J - \tau_N^J(\ell)) \end{bmatrix}, \tag{4.6}$$

and n is the accumulated noise across rounds.

By solving a least-squares problem:

$$\Delta^*(\ell) = (A^\top A)^{-1} A^\top \begin{bmatrix} x_{\text{agg}}^1(\ell) \\ \vdots \\ x_{\text{agg}}^J(\ell) \end{bmatrix}, \tag{4.7}$$

the server can estimate the local gradients $\Delta^*(\ell) = [\Delta_1(\ell), \ldots, \Delta_N(\ell)]^\top$. Once the gradients are recovered, any known gradient inversion attack (e.g., based on cosine similarity or label matching) can be applied to reconstruct the users' input data.

Chapter 5

The DynamicSecAgg Framework

The DynamicSecAgg Framework

DynamicSecAgg is a coordinate-hiding secure aggregation framework designed to preserve the privacy of both gradient values and their corresponding coordinates in federated learning. It extends the core ideas of TinySecAgg by introducing dynamic sparsity—allowing the number and choice of selected gradient coordinates to adapt during training. This enables the system to better balance communication cost, and model convergence.

Parameters

In each round t, the following parameters are involved:

- **Public parameters:**
  - User identifiers: $\{\alpha_i\}_{i\in[N]}$
  - Evaluation points: $\{\beta_n\}_{n\in[M+T]}$
  - Sparsity levels: $\{k_i^t\}_{i\in[N]}$, where k_i^t denotes the number of coordinates selected by user i in round t, with $k_{\min} \leq k_i^t \leq K_{\max}$
  - β_1 to β_M are used to encode true coordinate positions (shards).
  - β_{M+1} to β_{M+T} introduce structured randomness to protect against up to T colluding users.
- **Private (offline) inputs of user i:**
  - A random subset of $K_{\max}$ coordinate indices from $[d]$, fixed throughout training.
  - One-hot encoding shards for each of the $K_{\max}$ preselected coordinates:

$$a_k = [a_{k1} \| a_{k2} \| \cdots \| a_{kM}], \quad \text{where } a_{kn} \in \mathbb{F}_p^{d/M}$$

Example: If $d = 8$, $M = 4$, and $k = 5$, then the one-hot vector $a_5 = [0,0,0,0,1,0,0,0]^T$ is partitioned as:

$$a_{51} = [0,0], \quad a_{52} = [0,0], \quad a_{53} = [1,0], \quad a_{54} = [0,0]$$

- Randomness precomputed for each $k \in [K_{\max}]$:
  - Value mask: $r_{ik} \in \mathbb{F}_p$
  - Coordinate masks: $v_{ikn}, u_{ikn} \in \mathbb{F}_p^{d/M}$ for $n \in \{M+1, \ldots, M+T\}$

Offline Phase

For each user $i \in [N]$, the set of selected coordinates in round t is denoted by:

$$K_i^t = \{\ell \in [d] \ : \ b_i^t(\ell) = 1\},$$

where $b_i^t \in \{0,1\}^d$ is the binary sparsification mask indicating which gradient coordinates are selected for communication.

Demonstration. Suppose the model dimension is $d = 10$, and user i selects $K = 3$ coordinates in round t. Let the binary mask b_i^t be:

$$b_i^t = [0,1,0,0,1,0,0,1,0,0]^T$$

This means the non-zero entries of b_i^t occur at positions 2, 5, and 8. Therefore, the set of selected coordinates is:

$$K_i^t = \{2,5,8\}$$

Offline Randomness Generation

Each user $i \in [N]$ generates the following randomness in the offline phase, based on their selected coordinates:

- Let k_i^t denote the number of coordinates selected by user i in round t. The maximum allowed sparsity is $K_{\max}$.
- Random masks:

  $$\{r_{ik}\}_{k \in [K_{\max}]} \in \mathbb{F}_p$$

  where each r_{ik} is a uniformly random value used to mask the k-th coordinate in the preselected subset. At runtime, only the first k_i^t are used.
- Random vectors:

  $$\{v_{ikn},\ u_{ikn}\}_{k \in [K_{\max}],\, n \in \{M+1,\ldots,M+T\}} \in \mathbb{F}_p^{d/M}$$

  where v_{ikn} and u_{ikn} are random padding vectors used for polynomial encoding and mask cancellation. These are generated offline to ensure security against up to T colluding users. At runtime, only those corresponding to the selected k_i^t coordinates are used.

Shard-Based Encoding of Coordinates

Let $a_k \in \{0,1\}^d$ be a one-hot vector for coordinate k, and let it be partitioned into M equal-sized shards:

$$a_k = \begin{bmatrix} a_{k1}^T & \cdots & a_{kM}^T \end{bmatrix}^T \tag{5.1}$$

Each shard $a_{km} \in \{0,1\}^{d/M}$ corresponds to a contiguous block of the original vector a_k. This partitioning allows the encoded coordinate to be distributed across multiple sub-vectors, which are later embedded into polynomials during the encoding process.

Demonstration. Consider an example where the dimension is $d = 8$ and the number of shards is $M = 4$. Then each shard has size $d/M = 2$. Suppose we want to encode coordinate $k = 5$. The one-hot vector $a_5 \in \{0,1\}^8$ is:

$$a_5 = [0,0,0,0,1,0,0,0]^T$$

We partition a_5 into $M = 4$ shards of size 2:

$$\begin{aligned} a_{51} &= [0,0]^T \\ a_{52} &= [0,0]^T \\ a_{53} &= [1,0]^T \\ a_{54} &= [0,0]^T \end{aligned}$$

Thus,

$$a_5 = \begin{bmatrix} a_{51}^T & a_{52}^T & a_{53}^T & a_{54}^T \end{bmatrix}^T = [0,0,0,0,1,0,0,0]^T$$

This shard-wise representation is later used in constructing the encoding polynomial $\varphi_{ik}(\alpha)$, where each a_{km} contributes to a term in the polynomial.

To perform secure and coordinate-hiding aggregation, each user encodes their selected coordinates using polynomial interpolation over a set of public evaluation points.

Encoding Scheme

Let user i randomly select a fixed subset of $K_{\max}$ coordinates from the full gradient space $[d]$ during the offline phase. These selected coordinates are locally indexed as $k \in [K_{\max}]$, where each k represents the position within user i's private selection.

To hide the identity of each selected coordinate, user i constructs two polynomials—one for coordinate hiding and one for mask cancellation—for every $k \in [K_{\max}]$:

Coordinate-Hiding Polynomial

$$\boldsymbol{\varphi_{ik}(\alpha)}:$$

$$\varphi_{ik}(\alpha) = \sum_{n=1}^{M} a_{K_i^t(k),n} \cdot \prod_{n' \in [M+T] \setminus \{n\}} \frac{\alpha - \beta_{n'}}{\beta_n - \beta_{n'}} + \sum_{n=M+1}^{M+T} v_{ikn}^t \cdot \prod_{n' \in [M+T] \setminus \{n\}} \frac{\alpha - \beta_{n'}}{\beta_n - \beta_{n'}} \quad (5.2)$$

Mask-Canceling Polynomial

$$\boldsymbol{\psi_{ik}(\alpha)}:$$

$$\psi_{ik}(\alpha) = \sum_{n=1}^{M} a_{K_i^t(k),n} \cdot r_{ik}^t \cdot \prod_{n' \in [M+T] \setminus \{n\}} \frac{\alpha - \beta_{n'}}{\beta_n - \beta_{n'}} + \sum_{n=M+1}^{M+T} u_{ikn}^t \cdot \prod_{n' \in [M+T] \setminus \{n\}} \frac{\alpha - \beta_{n'}}{\beta_n - \beta_{n'}} \quad (5.3)$$

In the offline phase, all clients engage in a pairwise exchange of polynomial evaluations to enable secure aggregation. Specifically, for every selected coordinate k, user i evaluates both polynomials at each other user's evaluation point α_j and sends the results:

$$\{\varphi_{ik}(\alpha_j),\ \psi_{ik}(\alpha_j)\} \quad \text{to each user } j \in [N],\ j \neq i$$

Properties of the Polynomials

The polynomials $\varphi_{ik}(\alpha)$ and $\psi_{ik}(\alpha)$ are constructed using *Lagrange interpolation*, where each term is formed using a Lagrange basis polynomial.

The Lagrange basis polynomial $L_n(\alpha)$ for a given point β_n is defined as:

$$L_n(\alpha) = \prod_{\substack{n' \in [M+T] \\ n' \neq n}} \frac{\alpha - \beta_{n'}}{\beta_n - \beta_{n'}}$$

This construction ensures the following interpolation property:

$$L_n(\beta_{n'}) = \begin{cases} 1, & \text{if } n = n' \\ 0, & \text{otherwise} \end{cases}$$

With this basis, the polynomials $\varphi_{ik}(\alpha)$ and $\psi_{ik}(\alpha)$ satisfy the following key properties at the evaluation points $\{\beta_n\}_{n=1}^{M+T}$:

- At $\alpha = \beta_n$ for $n \leq M$:

$$\varphi_{ik}(\beta_n) = a_{K_i^t(k),n} \quad \text{(true coordinate shard)}$$

- At $\alpha = \beta_n$ for $n > M$:

$$\varphi_{ik}(\beta_n) = v_{ikn}^t \quad \text{(random noise)}$$

- At $\alpha = \beta_n$ for $n \leq M$:

$$\psi_{ik}(\beta_n) = a_{K_i^t(k),n} \cdot r_{ik}^t \quad \text{(scaled shard mask)}$$

- At $\alpha = \beta_n$ for $n > M$:

$$\psi_{ik}(\beta_n) = u_{ikn}^t \quad \text{(random noise)}$$

These properties ensure that the coordinate-shard structure is recoverable at positions β_1 through β_M, while all other positions inject structured randomness, protecting against collusion by up to T clients.

Online Phase

At the end of the offline phase, each client $i \in [N]$ holds the following set of information:

$$\left\{ \left(\varphi_{jk}(\alpha_i),\ \psi_{jk}(\alpha_i)\right) \mid j \in [N],\ j \neq i,\ k \in [k_{max}] \right\}$$

At the beginning of round t, each client computes its local score value $s_i^{(t)}$ based on its gradient information and sends it to the server, signed to ensure authenticity. The server collects $\{s_j^{(t)}\}_{j \in \mathcal{U}(t)}$ from all active users and broadcasts this set to all participants. This enables each client to perform score normalization consistently across the round. Each client then uses the normalized score value to compute its own sparsity level $k_i^{(t)}$ within the predefined bounds $[K_{\min}, K_{\max}]$. Now Each user i computes a masked version of its selected gradient entries:

$$\hat{x}_{ik}^t = x_i^t(K_i^t(k)) - r_{ik}^t, \quad \forall k \in [k_i] \tag{5.4}$$

These masked values $\hat{x}_{ik}^t$ are then broadcast to the other clients, one for each selected coordinate $k \in [k_i]$.

Then, each client i computes a local polynomial evaluation:

$$\phi(\alpha_i) = \sum_{j \in U(t)} \sum_{k \in [k_j]} \left(\hat{x}_{jk}^t \cdot \varphi_{jk}(\alpha_i) + \psi_{jk}(\alpha_i) \right) \tag{5.5}$$

The server collects $\phi(\alpha_i)$ from any $M + T$ users and reconstructs the global polynomial $\phi(\alpha)$ via Lagrange interpolation. It evaluates this polynomial at points

$\beta_1, \ldots, \beta_M$ to recover the aggregate gradient:

$$x_{\text{agg}}^t = \left[\phi^T(\beta_1)\ \phi^T(\beta_2)\ \cdots\ \phi^T(\beta_M)\right]^T \tag{5.6}$$

Finally, the global model is updated as:

$$w^{t+1} = w^t - \frac{1}{|U(t)|} f^{-1}(x_{\text{agg}}^t) \tag{5.7}$$

Details of the Finite Field Transformation

The local update $x_i^t(\ell)$ of each user i is converted to the finite field as follows:

$$x_i^t(\ell) \triangleq f\left(x_i^t(\ell)\right) \mod p \quad \forall \ell \in K_i^t \tag{5.8}$$

where $f(\cdot)$ is a stochastic rounding function, defined as:

$$f(x) \triangleq \begin{cases} \lfloor qx \rfloor \mod p & \text{with probability } 1 - (qx - \lfloor qx \rfloor) \\ \lfloor qx \rfloor + 1 \mod p & \text{otherwise} \end{cases}$$

Here, q is a tuning parameter that controls the rounding loss. The modulo operation maps both positive and negative integers into a finite field $\mathbb{F}_p$, using two's complement representation — where the first half of the field represents non-negative values, and the second half represents negative values.

After decoding the aggregated gradients, the server updates the global model as in Equation (5.7), where $f^{-1} : \mathbb{F}_p \to \mathbb{R}$ maps the finite field values back to the real domain:

$$f^{-1}(x) = \begin{cases} x/q & \text{if } 0 \le x < \frac{p-1}{2} \\ (x-p)/q & \text{if } \frac{p-1}{2} \le x \le p \end{cases}$$

Dynamic Sparsity Selection

Composite Contribution-Based k_i Selection. We propose a dynamic sparsity allocation method in which each client i computes its own sparsity level k_i based on a weighted combination of three factors:

- The ℓ_2 norm of the local gradient vector, $S_i^{\text{grad}} = \|\nabla w_i\|_2$
- The reduction in local loss after training, $S_i^{\text{loss}} = L_i^{\text{before}} - L_i^{\text{after}}$
- The standard deviation of the gradient, $S_i^{\text{std}} = \text{std}(\nabla w_i)$

Each client computes a raw score using a weighted sum of these components:

$$\text{score}_i = \alpha \cdot S_i^{\text{grad}} + \beta \cdot S_i^{\text{loss}} + \gamma \cdot S_i^{\text{std}} \tag{5.9}$$

The scores are then normalized across all clients in the round:

$$\text{norm\_score}_i = \frac{\text{score}_i - \min_j \text{score}_j}{\max_j \text{score}_j - \min_j \text{score}_j + \varepsilon} \tag{5.10}$$

where ε is a small constant to prevent division by zero.

Finally, the sparsity level k_i is determined as:

$$k_i = K_{\min} + (K_{\max} - K_{\min}) \cdot \text{norm\_score}_i \tag{5.11}$$

Implementation details:

- The weights α, β, and γ are tunable hyperparameters typically chosen within the range $[0, 1]$, with the constraint that $\alpha + \beta + \gamma = 1$
- **Gradient Norm:**

$$S_i^{\text{grad}} = \|\nabla w_i\|_2 = \sqrt{\sum_{j=1}^{d} \left(\nabla w_i^{(j)}\right)^2} \tag{5.12}$$

  where $\nabla w_i \in \mathbb{R}^d$ is the flattened local gradient or model update vector for client i.

- **Loss Reduction:**

$$S_i^{\text{loss}} = L_i^{\text{before}} - L_i^{\text{after}} \tag{5.13}$$

  where L_i^{before} and L_i^{after} are the local training losses measured before and after client i performs its local update.

- **Gradient Standard Deviation:**

$$S_i^{\text{std}} = \sqrt{\frac{1}{d}\sum_{j=1}^{d} \left(\nabla w_i^{(j)} - \mu_i\right)^2}, \quad \text{where } \mu_i = \frac{1}{d}\sum_{j=1}^{d} \nabla w_i^{(j)} \tag{5.14}$$

  is the standard deviation of the components of the gradient vector ∇w_i for client i.

Normalization of Local Contribution Metrics. To ensure that the individual score components remain within a bounded and comparable range, each client locally clips and normalizes its gradient-related metrics before constructing the overall score. Specifically:

- **Loss Reduction:**

$$S_i^{\text{loss}} = \frac{\Delta L_i + \log C}{2 \log C}, \quad \Delta L_i = L_i^{\text{before}} - L_i^{\text{after}} \tag{5.15}$$

where C is the number of classes. This maps the range $\Delta L_i \in [-\log C, \log C]$ into $[0, 1]$.

- **Gradient Norm:**

$$S_i^{\text{grad}} = \frac{\min\left(\|\nabla w_i\|_2, \tau\right)}{\tau} \tag{5.16}$$

where τ is a clipping threshold, typically selected from a predefined set (e.g., $\{10, 20, 50\}$) based on the range of gradient norms observed in the global model. For instance, if global gradient norms commonly fall within $[0, 10]$, then $\tau = 10$ is used; if they tend to lie in a higher range such as $[10, 20]$, a larger τ (e.g., 20) is chosen. This ensures stable normalization while retaining meaningful client-level variation.

- **Gradient Standard Deviation:**

$$S_i^{\text{std}} = \frac{\min\left(\text{std}(\nabla w_i), \tau\right)}{\tau} \tag{5.17}$$

using the same threshold τ as above to maintain consistency.

Chapter 6

Theoretical Analysis

Theorem 1 (Correctness)**.** *DynamicSecAgg ensures the correct recovery of the aggregate of sparsified gradients:*

$$x_{\text{agg}}^t = \sum_{j \in U(t)} x_j^t$$

from the messages of any set $U(t)$ of surviving users, provided that $|U(t)| \geq M + T$. Specifically, we have

$$H\left(\sum_{j \in U(t)} x_j^t \;\middle|\; \{\hat{x}_j^t\}_{j \in U(t)}, \{\phi(\alpha_j)\}_{j \in U(t)} \right) = 0$$

where x_j^t is the sparsified local gradient of user j, and $\hat{x}_j^t$, $\phi(\alpha_j)$ are the corresponding masked values and polynomial evaluations shared with the server.

Proof. We aim to show that DynamicSecAgg allows the server to correctly reconstruct the aggregate:

$$x_{\text{agg}}^t = \sum_{j \in U(t)} x_j^t$$

from the received masked gradients $\{\hat{x}_j^t\}$ and the evaluations $\{\phi(\alpha_j)\}$ of the encoding polynomial.

Let each user $j \in U(t)$ have sparsity level k_j^t, meaning it selects k_j^t coordinates from its gradient. Denote the ordered list of selected coordinates as K_j^t, and let $K_j^t(k)$ be the k-th selected coordinate.

Each user computes masked values:

$$\hat{x}_{jk}^t = x_j^t(K_j^t(k)) - r_{jk}^t$$

for $k \in [k_j^t]$, and defines two Lagrange interpolation polynomials $\varphi_{jk}(\cdot)$ and $\psi_{jk}(\cdot)$ based on the coordinate encoding and randomness.

Then, for each $i \in U(t)$, the following is sent:

$$\phi(\alpha_i) = \sum_{j \in U(t)} \sum_{k=1}^{k_j^t} \left(\hat{x}_{jk}^t \cdot \varphi_{jk}(\alpha_i) + \psi_{jk}(\alpha_i) \right)$$

Expanding this using the polynomial definitions:

$$\phi(\alpha_i) = \sum_{j\in U(t)} \sum_{k=1}^{k_j^t} \left[x_j^t(K_j^t(k)) \sum_{n=1}^{M} a_{K_j^t(k),n} \prod_{n'\in[M+T]\setminus\{n\}} \frac{\alpha_i - \beta_{n'}}{\beta_n - \beta_{n'}} + \sum_{n=M+1}^{M+T} \left(x_j^t(K_j^t(k)) v_{jkn}^t - r_{jk}^t v_{jkn}^t + u_{jkn}^t \right) \prod_{n'\in[M+T]\setminus\{n\}} \frac{\alpha_i - \beta_{n'}}{\beta_n - \beta_{n'}} \right]$$

This is a degree-$M+T-1$ polynomial in α, and since we have at least $M+T$ such evaluations, the server can interpolate and recover the full polynomial $\phi(\alpha)$.

Once the server reconstructs $\phi(\alpha)$, it evaluates the polynomial at shard points $\beta_1, \ldots, \beta_M$, obtaining:

$$\phi(\beta_n) = \sum_{j\in U(t)} \sum_{k=1}^{k_j^t} x_j^t(K_j^t(k)) \cdot a_{K_j^t(k),n}$$

where $a_{K_j^t(k),n}$ is the shard-assignment indicator for the coordinate.

Each $\phi(\beta_n)$ gives the aggregate over coordinates in shard n:

$$\phi(\beta_n) = \left[\sum_{j\in U(t)} x_j^t((n-1)\tfrac{d}{M}+1), \ldots, \sum_{j\in U(t)} x_j^t(n\tfrac{d}{M}) \right]^T$$

By concatenating all M shards, the server reconstructs the entire aggregated vector:

$$x_{\text{agg}}^t = \sum_{j\in U(t)} x_j^t$$

Thus, the entropy of the aggregate conditioned on the received information is zero:

$$H\left(\sum_{j\in U(t)} x_j^t \,\middle|\, \{\hat{x}_j^t\}_{j\in U(t)}, \{\phi(\alpha_j)\}_{j\in U(t)} \right) = 0$$

which completes the proof. □

Theorem 2 (Communication Complexity)**.** *DynamicSecAgg has per-user communication complexity of*

$$\mathcal{O}(k_{mean} + \tfrac{d}{M}) \quad \textit{(online)}, \qquad \mathcal{O}(K_{max} N \tfrac{d}{M}) \quad \textit{(offline)}$$

where k_{mean} denotes the average per-user sparsity level k_i^t in the online phase, and K_{max} is the maximum per-user sparsity level across all users used in offline setup.

Proof. We analyze the per-user communication in both the online and offline phases.

Online Phase. Each user $i \in [N]$ selects k_i^t coordinates and sends:

- $\mathcal{O}(k_i^t)$ symbols for broadcasting masked values $\hat{x}_{ik}^t$ for each $k \in [k_i^t]$,

- $\mathcal{O}(\frac{d}{M})$ symbols for sending vector $\phi(\alpha_i)$ to the server,
- $\mathcal{O}(1)$ symbol to broadcast the value of $score_i$ to the server.

Therefore, the total online communication per user is:

$$\mathcal{O}(k_i^t + \tfrac{d}{M}) \quad \text{(per user)}.$$

In the best case:

$$\mathcal{O}(K_{\min} + \tfrac{d}{M}).$$

In the average case:

$$\mathcal{O}(k_{\text{mean}} + \tfrac{d}{M}).$$

In the worst case:

$$\mathcal{O}(K_{\max} + \tfrac{d}{M}).$$

Offline Phase. In the offline stage, all users prepare interpolation points for the maximum possible number of coordinates, $K_{\max}$. Each user $i \in [N]$ sends:

- $\mathcal{O}(K_{\max} N \frac{d}{M})$ symbols for sharing $\varphi_{ik}(\alpha_j)$ with all other users,
- $\mathcal{O}(K_{\max} N \frac{d}{M})$ symbols for sharing $\psi_{ik}(\alpha_j)$ similarly.

Thus, the total offline communication per user is:

$$\mathcal{O}(K_{\max} N \tfrac{d}{M}).$$

Hence, DynamicSecAgg achieves the desired communication bounds under dynamic sparsity settings. Additionally, we can select $K_{\max}$ according to system feasibility and application-specific constraints, enabling flexible trade-offs between offline cost and maximum allowable sparsity. □

Theorem 3 (Computation Complexity with Dynamic Sparsity)**.** *The per-user computation complexity of TinySecAgg is*

$$\mathcal{O}\left(Nk_{mean}\tfrac{d}{M}\right) \quad \textit{(online)}, \qquad \mathcal{O}\left(Nk_{max}\tfrac{d}{M}\log^2(M+T)\log\log(M+T)\right) \quad \textit{(offline)}$$

where k_i^t is the dynamic sparsity level of user i, k_{mean} is the average sparsity, and k_{max} is the maximum sparsity across all users.

Proof. **Online Phase.** The online computation overhead of user $i \in [N]$ consists of:

- $\mathcal{O}(k_i^t)$ to compute the masked sparse gradient from (5.5),
- $\mathcal{O}\left(Nk_i^t\tfrac{d}{M}\right)$ to compute $\phi(\alpha_i)$ in (24).

Averaging over all users, the expected online complexity becomes:

$$\mathcal{O}\left(Nk_{\text{mean}}\frac{d}{M}\right).$$

Offline Phase. Interpolating a polynomial of degree γ, and evaluating it at γ points incurs a computational cost of $\gamma \log^2 \gamma \log\log \gamma$ (**?**). Therefore, the offline computation overhead of user $i \in [N]$ consists of:

- $\mathcal{O}\left(Nk_{\max}\frac{d}{M}\log^2(M+T)\log\log(M+T)\right)$ to compute $\varphi_{ik}(\alpha_j)$ in (20) for all $k \in [k_{\max}]$ and $j \in [N]$,
- $\mathcal{O}\left(Nk_{\max}\frac{d}{M}\log^2(M+T)\log\log(M+T)\right)$ to compute $\psi_{ik}(\alpha_j)$ in (21) for all $k \in [k_{\max}]$ and $j \in [N]$.

□

Framework	**Online**	**Offline**
SecAgg	$\mathcal{O}(N+d)$	$\mathcal{O}(N)$
SecAgg+	$\mathcal{O}(\log N + d)$	$\mathcal{O}(\log N)$
LightSecAgg	$\mathcal{O}(d+\frac{d}{M})$	$\mathcal{O}(Nd/M)$
TinySecAgg	$\mathcal{O}(K+\frac{d}{M})$	$\mathcal{O}(KNd/M)$
DynamicSecAgg	$\mathcal{O}(k_{\text{mean}}+\frac{d}{M})$	$\mathcal{O}(K_{\max}Nd/M)$

Table 6.1: Per-user communication complexity.

Theorem 4 (Privacy)**.** *DynamicSecAgg ensures information-theoretic privacy for both the sparsified gradient values and their coordinates against up to T colluding adversaries, for any set of surviving users $U(t)$ with $|U(t)| \geq M+T$, even when the users adopt dynamic sparsity patterns with per-user sparsity $k_i \leq K_{\max} \ll d$. Specifically, we have:*

$$I\left(\{x_i^t, K_i^t\}_{i\in[N]\setminus\mathcal{T}} \;\middle|\; \sum_{i\in U(t)} x_i^t, \{x_i^t, K_i^t\}_{i\in\mathcal{T}}, \mathcal{M}_{\mathcal{T}}^t, \mathcal{R}_{\mathcal{T}}^t\right) = 0$$

where:

- x_i^t *is the sparsified local gradient of user i,*
- K_i^t *is the set of coordinates selected by user i with* $|K_i^t| = k_i^t \leq K_{\max}$,
- $\mathcal{T} \subset [N]$ *is the set of adversarial users, with* $|\mathcal{T}| \leq T$,
- $\mathcal{M}_{\mathcal{T}}^t = \{\varphi_{ik}(\alpha_j), \psi_{ik}(\alpha_j)\}_{i\in[N], j\in\mathcal{T}, k\in[k_i^t]} \cup \{\phi(\alpha_i)\}_{i\in U(t)}$ *is the set of all messages received by the adversaries and server,*

- $\mathcal{R}_{\mathcal{T}}^t = \{r_{ik}^t\}_{i\in\mathcal{T},k\in[k_i]} \cup \{v_{ikn}^t, u_{ikn}^t\}_{i\in\mathcal{T},k\in[k_i^t],n\in\{M+1,\ldots,M+T\}}$ *is the randomness generated by the adversaries.*

Sketch. We define

$$m_{ijk}^t \triangleq \sum_{n=M+1}^{M+T} v_{ikn}^t \prod_{n'\in[M+T]\setminus\{n\}} \frac{\alpha_j - \beta_{n'}}{\beta_n - \beta_{n'}} \tag{6.1}$$

for all $i \in \mathcal{H},\ j \in \mathcal{T},\ k \in [k_i^t]$.

$$y_{ijk}^t \triangleq \sum_{n=M+1}^{M+T} u_{ikn}^t \prod_{n'\in[M+T]\setminus\{n\}} \frac{\alpha_j - \beta_{n'}}{\beta_n - \beta_{n'}} \tag{6.2}$$

$$\gamma_{jn} \triangleq \prod_{n'\in[M+T]\setminus\{n\}} \frac{\alpha_j - \beta_{n'}}{\beta_n - \beta_{n'}} \tag{6.3}$$

Then,

$$\begin{bmatrix} m_{i_1jk}^t & \cdots & m_{i_Tjk}^t \end{bmatrix} = \begin{bmatrix} v_{i,k,M+1}^t & \cdots & v_{i,k,M+T}^t \end{bmatrix} X, \tag{6.4}$$

$$\begin{bmatrix} y_{i_1jk}^t & \cdots & y_{i_Tjk}^t \end{bmatrix} = \begin{bmatrix} u_{i,k,M+1}^t & \cdots & u_{i,k,M+T}^t \end{bmatrix} X, \tag{6.5}$$

where

$$X \triangleq \begin{bmatrix} \gamma_{1,M+1} & \cdots & \gamma_{T,M+1} \\ \vdots & \ddots & \vdots \\ \gamma_{1,M+T} & \cdots & \gamma_{T,M+T} \end{bmatrix}. \tag{6.6}$$

Note that X is a $T \times T$ MDS (Maximum Distance Separable) matrix (and hence invertible) due to the MDS property of Lagrange coefficients.

As such, one can compute $\{v_{ikn}^t, u_{ikn}^t\}_{i\in\mathcal{H},\ n\in\{M+1,\ldots,M+T\},\ k\in[k_i^t]}$ given

$$\{m_{ijk}^t\}_{i\in\mathcal{H},\ j\in\mathcal{T},\ k\in[k_i^t]}, \quad \text{and} \quad \{y_{ijk}^t\}_{i\in\mathcal{H},\ j\in\mathcal{T},\ k\in[k_i^t]}$$

Let's we consider a worst-case setting with $|U(t)| = M + T$ surviving users, and adversaries observing all communication. The privacy condition becomes:

$$I\left(\{x_i^t, K_i^t\}_{i\in\mathcal{H}} \;\middle|\; \{\hat{x}_{ik}^t\}_{i\in[N],k\in[k_i^t]},\ \{\varphi_{ik}(\alpha_j)\}_{i\in[N],j\in\mathcal{T},k\in[k_i^t]},\ \{\psi_{ik}(\alpha_j)\}_{i\in[N],j\in\mathcal{T},k\in[k_i^t]},\ \{\phi(\alpha_i)\}_{i\in U(t)} \right.$$

$$\left. \sum_{i\in U(t)} x_i^t,\ \{x_i^t, K_i^t\}_{i\in\mathcal{T}},\ \{r_{ik}^t\}_{i\in\mathcal{T},k\in[k_i^t]},\ \{v_{ikn}^t, u_{ikn}^t\}_{i\in\mathcal{T},k\in[k_i^t],n\in\{M+1,\ldots,M+T\}} \right) = 0$$

To analyze the information leakage, we use the standard identity for conditional mutual information:

$$I(A; B \mid C) = H(B \mid C) - H(B \mid A, C),$$

which quantifies how much information the variable set B reveals about A, condi-

tioned on knowing C. In our case:

- $A = \{x_i^t, K_i^t\}_{i\in\mathcal{H}}$ is the set of honest users' private data (sparsified gradients and selected coordinates),
- $B = \left\{\hat{x}_{ik}^t, \varphi_{ik}(\alpha_j), \psi_{ik}(\alpha_j), \phi(\alpha_i)\right\}$ is the complete view of the adversaries and the server,
- $C = \left(\sum_{i\in U(t)} x_i^t, \{x_i^t, K_i^t\}_{i\in\mathcal{T}}, \mathcal{R}_{\mathcal{T}}^t\right)$ captures what the adversaries already know.

$$
\begin{aligned}
&I\Big(\{x_i^t, K_i^t\}_{i\in\mathcal{H}} \;\Big|\; \{\hat{x}_{ik}^t\}_{i\in[N],k\in[k_i^t]},\ \{\varphi_{ik}(\alpha_j)\}_{i\in[N],j\in\mathcal{T},k\in[k_i^t]},\ \{\psi_{ik}(\alpha_j)\}_{i\in[N],j\in\mathcal{T},k\in[k_i^t]},\ \{\phi(\alpha_i)\}_{i\in U(t)} \\
&\quad\Big|\ \sum_{i\in U(t)} x_i^t,\ \{x_i^t, K_i^t\}_{i\in\mathcal{T}},\ \{r_{ik}^t\}_{i\in\mathcal{T},k\in[k_i^t]},\ \{v_{ikn}^t, u_{ikn}^t\}_{i\in\mathcal{T},k\in[k_i^t],n\in\{M+1,\ldots,M+T\}}\Big) \\
&= H\Big(\{\hat{x}_{ik}^t\}_{i\in[N],k\in[k_i^t]},\ \{\varphi_{ik}(\alpha_j)\}_{i\in[N],j\in\mathcal{T},k\in[k_i^t]},\ \{\psi_{ik}(\alpha_j)\}_{i\in[N],j\in\mathcal{T},k\in[k_i^t]},\ \{\phi(\alpha_i)\}_{i\in U(t)} \;\Big| \\
&\quad\sum_{i\in U(t)} x_i^t,\ \{x_i^t, K_i^t\}_{i\in\mathcal{T}},\ \{r_{ik}^t\}_{i\in\mathcal{T},k\in[k_i^t]},\ \{v_{ikn}^t, u_{ikn}^t\}_{i\in\mathcal{T},k\in[k_i^t],n\in\{M+1,\ldots,M+T\}}\Big) \\
&\quad- H\Big(\{\hat{x}_{ik}^t\}_{i\in[N],k\in[k_i^t]},\ \{\varphi_{ik}(\alpha_j)\}_{i\in[N],j\in\mathcal{T},k\in[k_i^t]},\ \{\psi_{ik}(\alpha_j)\}_{i\in[N],j\in\mathcal{T},k\in[k_i^t]},\ \{\phi(\alpha_i)\}_{i\in U(t)} \;\Big| \\
&\quad\sum_{i\in U(t)} x_i^t,\ \{x_i^t, K_i^t\}_{i\in[N]},\ \{r_{ik}^t\}_{i\in\mathcal{T},k\in[k_i^t]},\ \{v_{ikn}^t, u_{ikn}^t\}_{i\in\mathcal{T},k\in[k_i^t],n\in\{M+1,\ldots,M+T\}}\Big) \quad (T4.1)
\end{aligned}
$$

We now simplify the second term in Equation (T4.1)

$$
\begin{aligned}
&H\Big(\{\hat{x}_{ik}^t\}_{i\in[N],k\in[k_i^t]},\ \{\varphi_{ik}(\alpha_j)\}_{i\in[N],j\in\mathcal{T},k\in[k_i^t]},\ \{\psi_{ik}(\alpha_j)\}_{i\in[N],j\in\mathcal{T},k\in[k_i^t]},\ \{\phi(\alpha_i)\}_{i\in U(t)} \;\Big| \\
&\quad\sum_{i\in U(t)} x_i^t,\ \{x_i^t, K_i^t\}_{i\in[N]},\ \{r_{ik}^t\}_{i\in\mathcal{T},k\in[k_i^t]},\ \{v_{ikn}^t, u_{ikn}^t\}_{i\in\mathcal{T},k\in[k_i^t],n\in\{M+1,\ldots,M+T\}}\Big)
\end{aligned}
$$

Assume the inputs $\{x_i^t, K_i^t\}_{i\in[N]}$ and all adversarial randomness are known, every message such as $\hat{x}_{ik}^t$, $\varphi_{ik}(\alpha_j)$, $\psi_{ik}(\alpha_j)$, and $\phi(\alpha_i)$ becomes a deterministic function—except for the randomness r_{ik}^t generated by the honest users. Therefore, the entropy of the observed messages reduces to the entropy of the honest users' random masks. Therefore

$$
\begin{aligned}
&= H\Big(\{r_{ik}^t\}_{i\in\mathcal{H},k\in[k_i^t]},\ \{\varphi_{ik}(\alpha_j)\}_{i\in[N],j\in\mathcal{T},k\in[k_i^t]},\ \{\psi_{ik}(\alpha_j)\}_{i\in[N],j\in\mathcal{T},k\in[k_i^t]},\ \{\phi(\alpha_i)\}_{i\in U(t)} \;\Big| \\
&\quad\sum_{i\in U(t)} x_i^t,\ \{x_i^t, K_i^t\}_{i\in[N]},\ \{r_{ik}^t\}_{i\in\mathcal{T},k\in[k_i^t]},\ \{v_{ikn}^t, u_{ikn}^t\}_{i\in\mathcal{T},k\in[k_i^t],n\in\{M+1,\ldots,M+T\}}\Big) \quad (T4.2)
\end{aligned}
$$

We apply the chain rule of entropy, which states:

$$H(A, B \mid C) = H(A \mid C) + H(B \mid A, C).$$

$$\begin{aligned} &= H\Big(\{\varphi_{ik}(\alpha_j)\}_{i\in[N], j\in\mathcal{T}, k\in[k_i^t]},\ \{\psi_{ik}(\alpha_j)\}_{i\in[N], j\in\mathcal{T}, k\in[k_i^t]},\ \{\phi(\alpha_i)\}_{i\in U(t)} \ \Big| \\ &\quad \sum_{i\in U(t)} x_i^t,\ \{x_i^t, K_i^t\}_{i\in[N]},\ \{r_{ik}^t\}_{i\in[N], k\in[k_i^t]},\ \{v_{ikn}^t, u_{ikn}^t\}_{i\in\mathcal{T}, k\in[k_i^t], n\in\{M+1,\ldots,M+T\}} \Big) \\ &\quad + H\Big(\{r_{ik}^t\}_{i\in\mathcal{H}, k\in[k_i^t]} \ \Big| \ \sum_{i\in U(t)} x_i^t,\ \{x_i^t, K_i^t\}_{i\in[N]},\ \{r_{ik}^t\}_{i\in\mathcal{T}, k\in[k_i^t]},\ \{v_{ikn}^t, u_{ikn}^t\}_{i\in\mathcal{T}, k\in[k_i^t], n\in\{M+1,\ldots,M+T\}} \Big) \end{aligned} \tag{T4.3}$$

$$\begin{aligned} &= H\Big(\{\varphi_{ik}(\alpha_j)\}_{i\in[N],\ j\in\mathcal{T},\ k\in[k_i^t]},\ \{\psi_{ik}(\alpha_j)\}_{i\in[N],\ j\in\mathcal{T},\ k\in[k_i^t]},\ \{\phi(\alpha_i)\}_{i\in U(t)} \ \Big| \ \sum_{i\in U(t)} x_i^t,\ \{x_i^t, K_i^t\}_{i\in[N]}, \\ &\quad \{r_{ik}^t\}_{i\in[N],\ k\in[k_i^t]},\ \{v_{ikn}^t,\ u_{ikn}^t\}_{i\in\mathcal{T},\ k\in[k_i^t],\ n\in\{M+1,\ldots,M+T\}} \Big) \\ &\quad + H\Big(\{r_{ik}^t\}_{i\in\mathcal{H},\ k\in[k_i^t]} \Big) \end{aligned} \tag{T4.4}$$

$$\begin{aligned} &= H\Big(\left\{m_{ijk}^t\right\}_{i\in\mathcal{H},\ j\in\mathcal{T},\ k\in[k_i^t]},\ \{\psi_{ik}(\alpha_j)\}_{i\in[N],\ j\in\mathcal{T},\ k\in[k_i^t]},\ \{\phi(\alpha_i)\}_{i\in U(t)} \ \Big| \\ &\quad \sum_{i\in U(t)} x_i^t,\ \{x_i^t, K_i^t\}_{i\in[N]},\ \{r_{ik}^t\}_{i\in[N],\ k\in[k_i^t]},\ \{v_{ikn}^t, u_{ikn}^t\}_{i\in\mathcal{T},\ k\in[k_i^t],\ n\in\{M+1,\ldots,M+T\}} \Big) \\ &\quad + H\Big(\{r_{ik}^t\}_{i\in\mathcal{H},\ k\in[k_i^t]} \Big) \end{aligned} \tag{T4.5}$$

m_{ijk}^t denotes the masked term embedded inside the polynomial evaluation sent from honest user i to adversarial user j for coordinate k, where $i \in \mathcal{H}$, $j \in \mathcal{T}$, and $k \in [k_i^t]$.

$$= H\Big(\{\psi_{ik}(\alpha_j)\}_{i\in[N],\, j\in\mathcal{T},\, k\in[k_i^t]},\ \{\phi(\alpha_i)\}_{i\in U(t)} \ \Big|\ \sum_{i\in U(t)} x_i^t,\ \{x_i^t, K_i^t\}_{i\in[N]},$$
$$\{r_{ik}^t\}_{i\in[N],\, k\in[k_i^t]},\ \{v_{ikn}^t, u_{ikn}^t\}_{i\in\mathcal{T},\, k\in[k_i^t],\, n\in\{M+1,\ldots,M+T\}},\ \Big\{m_{ijk}^t\Big\}_{i\in\mathcal{H},\, j\in\mathcal{T},\, k\in[k_i^t]} \Big)$$
$$+ H\Big(\Big\{m_{ijk}^t\Big\}_{i\in\mathcal{H},\, j\in\mathcal{T},\, k\in[k_i^t]} \Big) + H\Big(\{r_{ik}^t\}_{i\in\mathcal{H},\, k\in[k_i^t]} \Big) \tag{T4.6}$$

$$= H\Big(\Big\{y_{ijk}^t\Big\}_{i\in\mathcal{H},\, j\in\mathcal{T},\, k\in[k_i^t]},\ \{\phi(\alpha_i)\}_{i\in U(t)} \ \Big|\ \sum_{i\in U(t)} x_i^t,\ \{x_i^t, K_i^t\}_{i\in[N]},$$
$$\{r_{ik}^t\}_{i\in[N],\, k\in[k_i^t]},\ \{v_{ikn}^t, u_{ikn}^t\}_{i\in\mathcal{T},\, k\in[k_i^t],\, n\in\{M+1,\ldots,M+T\}},\ \Big\{m_{ijk}^t\Big\}_{i\in\mathcal{H},\, j\in\mathcal{T},\, k\in[k_i^t]} \Big)$$
$$+ H\Big(\Big\{m_{ijk}^t\Big\}_{i\in\mathcal{H},\, j\in\mathcal{T},\, k\in[k_i^t]} \Big) + H\Big(\{r_{ik}^t\}_{i\in\mathcal{H},\, k\in[k_i^t]} \Big) \tag{T4.7}$$

$$= H\Big(\{\phi(\alpha_i)\}_{i\in U(t)} \ \Big|\ \sum_{i\in U(t)} x_i^t,\ \{x_i^t, K_i^t\}_{i\in[N]},\ \{r_{ik}^t\}_{i\in[N],\, k\in[k_i^t]},$$
$$\{v_{ikn}^t, u_{ikn}^t\}_{i\in\mathcal{T},\, k\in[k_i^t],\, n\in\{M+1,\ldots,M+T\}},\ \Big\{m_{ijk}^t\Big\}_{i\in\mathcal{H},\, j\in\mathcal{T},\, k\in[k_i^t]},\ \Big\{y_{ijk}^t\Big\}_{i\in\mathcal{H},\, j\in\mathcal{T},\, k\in[k_i^t]} \Big)$$
$$+ H\Big(\Big\{y_{ijk}^t\Big\}_{i\in\mathcal{H},\, j\in\mathcal{T},\, k\in[k_i^t]} \Big) + H\Big(\Big\{m_{ijk}^t\Big\}_{i\in\mathcal{H},\, j\in\mathcal{T},\, k\in[k_i^t]} \Big) + H\Big(\{r_{ik}^t\}_{i\in\mathcal{H},\, k\in[k_i^t]} \Big) \tag{T4.8}$$

$$= 0 + H\Big(\Big\{y_{ijk}^t\Big\}_{i\in\mathcal{H},\, j\in\mathcal{T},\, k\in[k_i^t]} \Big) + H\Big(\Big\{m_{ijk}^t\Big\}_{i\in\mathcal{H},\, j\in\mathcal{T},\, k\in[k_i^t]} \Big)$$
$$+ H\Big(\{r_{ik}^t\}_{i\in\mathcal{H},\, k\in[k_i^t]} \Big) \tag{T4.9}$$

$$= H\Big(\{v_{ikn}^t\}_{i\in\mathcal{H},\, k\in[k_i^t],\, n\in\{M+1,\ldots,M+T\}} \Big) + H\Big(\{u_{ikn}^t\}_{i\in\mathcal{H},\, k\in[k_i^t],\, n\in\{M+1,\ldots,M+T\}} \Big)$$
$$+ H\Big(\{r_{ik}^t\}_{i\in\mathcal{H},\, k\in[k_i^t]} \Big) \tag{T4.10}$$

$$= \sum_{i\in\mathcal{H}} T\cdot k_i^t\cdot\frac{d}{M}\log p + \sum_{i\in\mathcal{H}} T\cdot k_i^t\cdot\frac{d}{M}\log p + \sum_{i\in\mathcal{H}} k_i^t\cdot\log p \tag{T4.11}$$

Next, the first term in (T4.1) can be bounded as follows:

$$H\Big(\{\bar{x}_{ik}^t\}_{i\in[N],\, k\in[k_i^t]},\ \{\varphi_{ik}(\alpha_j)\}_{i\in[N],\, j\in\mathcal{T},\, k\in[k_i^t]},\ \{\psi_{ik}(\alpha_j)\}_{i\in[N],\, j\in\mathcal{T},\, k\in[k_i^t]},\ \{\phi(\alpha_i)\}_{i\in U(t)}$$
$$\Big|\ \sum_{i\in U(t)} x_i^t,\ \{x_i^t, K_i^t\}_{i\in\mathcal{T}},\ \{r_{ik}^t\}_{i\in\mathcal{T},\, k\in[k_i^t]},\ \{v_{ikn}^t, u_{ikn}^t\}_{i\in\mathcal{T},\, k\in[k_i^t],\, n\in\{M+1,\ldots,M+T\}} \Big)$$
$$= H\Big(\{\bar{x}_{ik}^t\}_{i\in\mathcal{H},\, k\in[k_i^t]},\ \{\varphi_{ik}(\alpha_j)\}_{i\in\mathcal{H},\, j\in\mathcal{T},\, k\in[k_i^t]},\ \{\psi_{ik}(\alpha_j)\}_{i\in\mathcal{H},\, j\in\mathcal{T},\, k\in[k_i^t]},\ \{\phi(\alpha_i)\}_{i\in U(t)}$$
$$\Big|\ \sum_{i\in U(t)} x_i^t,\ \{x_i^t, K_i^t\}_{i\in\mathcal{T}},\ \{r_{ik}^t\}_{i\in\mathcal{T},\, k\in[k_i^t]},\ \{v_{ikn}^t, u_{ikn}^t\}_{i\in\mathcal{T},\, k\in[k_i^t],\, n\in\{M+1,\ldots,M+T\}} \Big) \tag{T4.12}$$

$$\underbrace{H(A,B,C,D \mid E)}_{\text{joint conditional entropy}} \leq \underbrace{H(A \mid E) + H(B \mid E) + H(C \mid E) + H(D \mid E)}_{\text{subadditivity of entropy}}$$
$$\leq H(A) + H(B) + H(C) + H(D \mid E) \tag{6.7}$$

Using the subadditivity of entropy, this is upper bounded by:

$$\leq H\Big(\{\bar{x}_{ik}^t\}_{i\in\mathcal{H},\,k\in[k_i^t]}\Big) + H\Big(\{\varphi_{ik}(\alpha_j)\}_{i\in\mathcal{H},\,j\in\mathcal{T},\,k\in[k_i^t]}\Big) + H\Big(\{\psi_{ik}(\alpha_j)\}_{i\in\mathcal{H},\,j\in\mathcal{T},\,k\in[k_i^t]}\Big)$$
$$+ H\Big(\{\phi(\alpha_i)\}_{i\in U(t)} \;\Big|\; \sum_{i\in U(t)} x_i^t,\ \{r_{ik}^t\}_{i\in\mathcal{T},\,k\in[k_i^t]},\ \{x_i^t, K_i^t\}_{i\in\mathcal{T}},\ \{v_{ikn}^t, u_{ikn}^t\}_{i\in\mathcal{T},\,k\in[k_i^t],\,n\in\{M+1,\ldots,M+T\}},$$
$$\{\bar{x}_{ik}^t\}_{i\in\mathcal{H},\,k\in[k_i^t]},\ \{\varphi_{ik}(\alpha_j)\}_{i\in\mathcal{H},\,j\in\mathcal{T},\,k\in[k_i^t]},\ \{\psi_{ik}(\alpha_j)\}_{i\in\mathcal{H},\,j\in\mathcal{T},\,k\in[k_i^t]}\Big) \tag{T4.13}$$

$$= H\left(\{\bar{x}_{ik}^t\}_{i\in\mathcal{H},\,k\in[k_i^t]}\right) + H\left(\{\varphi_{ik}(\alpha_j)\}_{i\in\mathcal{H},\,j\in\mathcal{T},\,k\in[k_i^t]}\right) + H\left(\{\psi_{ik}(\alpha_j)\}_{i\in\mathcal{H},\,j\in\mathcal{T},\,k\in[k_i^t]}\right)$$
$$+ H\Big(\{\phi(\beta_n)\}_{n\in[M]},\ \{\phi(\alpha_i)\}_{i\in\mathcal{T}} \;\Big|\; \sum_{i\in U(t)} x_i^t,\ \{x_i^t, K_i^t\}_{i\in\mathcal{T}},\ \{r_{ik}^t\}_{i\in\mathcal{T},\,k\in[k_i^t]},$$
$$\{v_{ikn}^t, u_{ikn}^t\}_{i\in\mathcal{T},\,k\in[k_i^t],\,n\in\{M+1,\ldots,M+T\}},\ \{\bar{x}_{ik}^t\}_{i\in\mathcal{H},\,k\in[k_i^t]},\ \{\varphi_{ik}(\alpha_j)\}_{i\in\mathcal{H},\,j\in\mathcal{T},\,k\in[k_i^t]},$$
$$\{\psi_{ik}(\alpha_j)\}_{i\in\mathcal{H},\,j\in\mathcal{T},\,k\in[k_i^t]}\Big) \tag{T4.14}$$

$$= H\left(\{\bar{x}_{ik}^t\}_{i\in\mathcal{H},\,k\in[k_i^t]}\right) + H\left(\{\varphi_{ik}(\alpha_j)\}_{i\in\mathcal{H},\,j\in\mathcal{T},\,k\in[k_i^t]}\right) + H\left(\{\psi_{ik}(\alpha_j)\}_{i\in\mathcal{H},\,j\in\mathcal{T},\,k\in[k_i^t]}\right) + 0 \tag{T4.15}$$

$$\leq \sum_{i\in\mathcal{H}} k_i^t \log p + \sum_{i\in\mathcal{H}}\sum_{j\in\mathcal{T}} \frac{k_i^t d}{M}\log p + \sum_{i\in\mathcal{H}}\sum_{j\in\mathcal{T}} \frac{k_i^t d}{M}\log p$$
$$= \sum_{i\in\mathcal{H}} k_i^t \cdot \log p + \sum_{i\in\mathcal{H}} T\cdot k_i^t \cdot \frac{d}{M}\log p + \sum_{i\in\mathcal{H}} T\cdot k_i^t \cdot \frac{d}{M}\log p \tag{T4.16}$$

Equation (T4.12) holds because the masked gradients $\{\bar{x}_{ik}^t\}_{i\in\mathcal{T},\,k\in[k_i^t]}$ are completely determined by the adversarial users' gradients, selected indices, and local random masks. Thus, there is no remaining uncertainty.

Equation (T4.13) follows from the *chain rule of entropy* and the fact that *conditioning cannot increase entropy*.

Equation (T4.14) is justified by observing that $\{\phi(\alpha_i)\}_{i\in U(t)}$ are evaluations of a polynomial $\phi(\alpha)$ of degree $M+T-1$. Since we have $M+T$ evaluations, polynomial interpolation ensures the polynomial can be uniquely reconstructed.

This implies a bijective mapping between the interpolation points $\{\phi(\beta_n)\}_{n\in[M]}$ and

$\{\phi(\alpha_i)\}_{i\in[T]}$, and the full set of local evaluations $\{\phi(\alpha_i)\}_{i\in U(t)}$.
Furthermore, there is no uncertainty in $\{\phi(\beta_n)\}_{n\in[M]}$ given the sum of user gradients $\sum_{i\in U(t)} x_i^t$, as established in Equation (5.6).
Similarly, the values $\{\phi(\alpha_i)\}_{i\in[T]}$ are fully determined by the adversarial users' gradients, index sets, random masks, shared randomness, and the observed messages. Therefore, Equation (T4.15) holds.
Finally, Equation (T4.16) follows because the entropy is maximized when the underlying random variables are uniformly distributed.
Finally, Equation (T4.16) holds since the uniform distribution maximizes entropy. By combining Equations (T4.11), (T4.16), and (T4.1), we get:

$$
\begin{aligned}
&I\Big(\{x_i^t, K_i^t\}_{i\in\mathcal{H}} \;\Big|\; \{\bar{x}_{ik}^t\}_{i\in[N],\,k\in[k_i^t]},\ \{\varphi_{ik}(\alpha_j)\}_{i\in[N],\,j\in\mathcal{T},\,k\in[k_i^t]},\ \{\psi_{ik}(\alpha_j)\}_{i\in[N],\,j\in\mathcal{T},\,k\in[k_i^t]},\\
&\qquad \{\phi(\alpha_i)\}_{i\in U(t)} \;\Big|\; \sum_{i\in U(t)} x_i^t,\ \{x_i^t, K_i^t\}_{i\in\mathcal{T}},\ \{r_{ik}^t\}_{i\in\mathcal{T},\,k\in[k_i^t]},\ \{v_{ikn}^t, u_{ikn}^t\}_{i\in\mathcal{T},\,k\in[k_i^t],\,n\in\{M+1,\dots,M+T\}}\Big)\\
&\le \sum_{i\in\mathcal{H}} k_i^t \log p + \sum_{i\in\mathcal{H}} \frac{T\cdot k_i^t\cdot d}{M}\log p + \sum_{i\in\mathcal{H}} \frac{T\cdot k_i^t\cdot d}{M}\log p\\
&\quad - \sum_{i\in\mathcal{H}} \frac{T\cdot k_i^t\cdot d}{M}\log p - \sum_{i\in\mathcal{H}} \frac{T\cdot k_i^t\cdot d}{M}\log p - \sum_{i\in\mathcal{H}} k_i^t \log p\\
&= 0
\end{aligned}
\tag{T4.17}
$$

Then, from Equation (T4.17) and the non-negativity of mutual information, we conclude:

$$
\begin{aligned}
&I\Big(\{x_i^t, K_i^t\}_{i\in\mathcal{H}} \;\Big|\; \{\bar{x}_{ik}^t\}_{i\in[N],\,k\in[k_i^t]},\ \{\varphi_{ik}(\alpha_j)\}_{\substack{i\in[N],\\ j\in\mathcal{T},\\ k\in[k_i^t]}},\ \{\psi_{ik}(\alpha_j)\}_{\substack{i\in[N],\\ j\in\mathcal{T},\\ k\in[k_i^t]}},\\
&\qquad \{\phi(\alpha_i)\}_{i\in U(t)} \;\Big|\; \sum_{i\in U(t)} x_i^t,\ \{x_i^t, K_i^t\}_{i\in\mathcal{T}},\ \{r_{ik}^t\}_{i\in\mathcal{T},\,k\in[k_i^t]},\\
&\qquad \{v_{ikn}^t, u_{ikn}^t\}_{\substack{i\in\mathcal{T},\\ k\in[k_i^t],\\ n\in\{M+1,\dots,M+T\}}}\Big) = 0
\end{aligned}
\tag{T4.18}
$$

which concludes the proof.

□

Chapter 7

Performance Evaluation

7.1 Experimental Setup

This section outlines the practical configuration used to compare *Dynamic Sparsity* with *Constant Sparsity* in a federated learning setting.

7.1.1 Federated Learning Configuration

The following experimental parameters are used throughout:

- **Number of Clients:** 10
- **Epochs per Client:** 5
- **Batch Size:** 64
- **Learning Rate:** 0.005
- **Shard Size:** $M = 40$

Each client trains locally for 5 epochs per round and communicates with the server using model update vectors. The system follows the federated averaging (`FedAvg`) protocol.

7.1.2 Dataset: MNIST

We utilize the MNIST dataset, consisting of 60,000 training and 10,000 testing grayscale images of handwritten digits (0–9), each of size 28×28. MNIST is a standard benchmark in federated learning due to its simplicity and clarity in assessing sparsity techniques.

7.1.3 Data Distribution: IID vs Non-IID

IID (Independent and Identically Distributed): In the IID setting, the training data is randomly and equally distributed among all 10 clients. This ensures that each client has a representative subset of the global data distribution, including all digit classes in approximately equal proportions.

Non-IID (NIID - Non-Identically Distributed): In the Non-IID configuration, the training data is partitioned such that each client receives data corresponding to a limited number of digit classes. This simulates real-world scenarios where users' local data distributions vary significantly. This setup introduces statistical heterogeneity, making model convergence and generalization more challenging.

7.1.4 Sparsity Parameters

To control communication efficiency, sparsity is applied to the model updates. Let d denote the total dimensionality of the model's update vector (i.e., the total number of trainable parameters).

The following sparsity bounds are used:

- **Maximum Sparsity:** $k_{\max} = 0.09d$
- **Minimum Sparsity:** $k_{\min} = 0.01d$
- **TinySecAgg Baseline:** A constant sparsity of $0.05d$

These values define the range within which dynamic sparsity strategies adaptively select the number of active dimensions per round. The baseline method, TinySecAgg, uses a fixed sparsity of 5% of the model dimension d, and serves as the comparison point for evaluating the benefits of dynamic sparsity.

7.1.5 Local Normalization Parameter (τ)

We set the local normalization parameter to $\tau = 10$ in all experiments.

7.2 Results and Analysis: IID Setting

This section presents the performance evaluation of various dynamic sparsity strategies compared to constant sparsity under an IID (Independent and Identically Distributed) data partitioning scheme.

7.2.1 Comparison: Constant vs Dynamic Sparsity

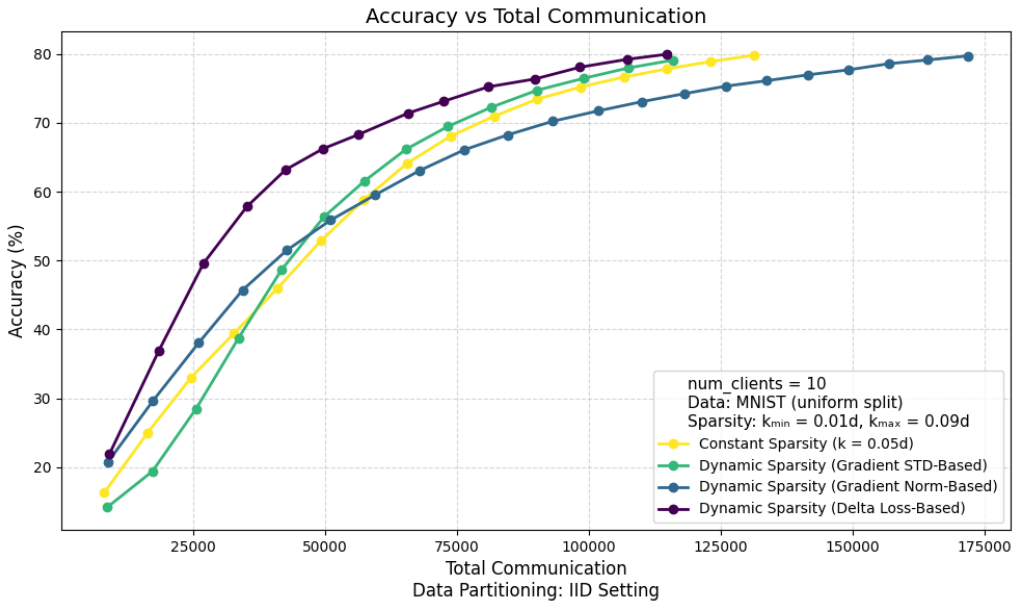

Figure 7.1: **Delta Loss-Based** sparsity achieves the best performance, surpassing 80% accuracy overall and reaching 65% accuracy early with minimal communication. **Gradient STD-Based** performs similarly to **Fixed Sparsity** (e.g., $k = 0.05d$), indicating its limited influence on dynamic selection under IID data distribution. **Norm-Based** sparsity achieves faster initial gains, reaching 50% accuracy efficiently, but plateaus around 78–79%. In contrast, delta-based sparsity continues to improve and dominates in later rounds. All dynamic strategies eventually outperform fixed sparsity baselines.

7.2.2 Effect of Different Dynamic Weight Combinations

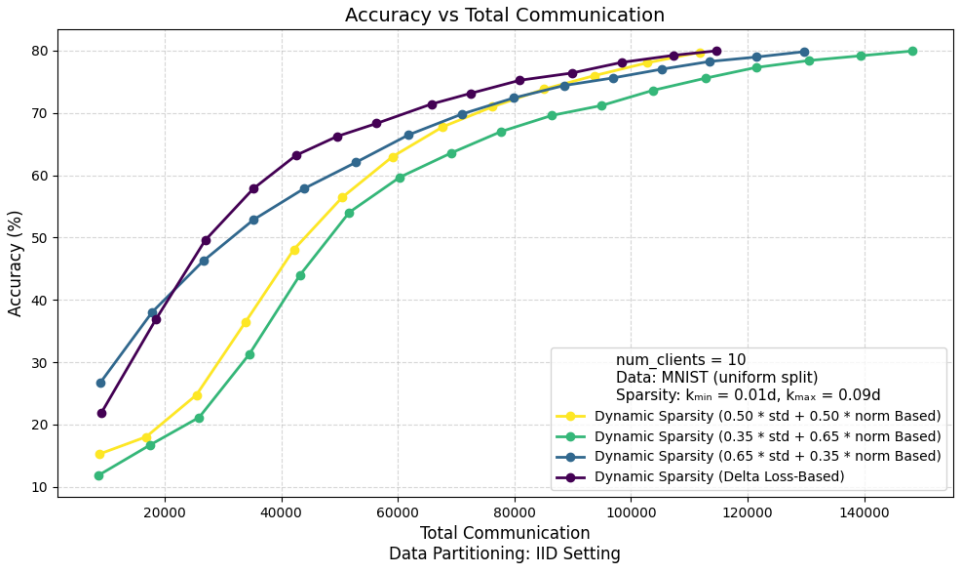

Figure 7.2: Comparison of hybrid dynamic sparsity strategies in IID setting. **Delta Loss-Based** dominates throughout, achieving **greater 60% accuracy** by **40k communication units** and maintaining lead. Hybrid methods (**0.65*std+0.35*norm**) show competitive early progress but plateau near **55% accuracy**, while other hybrids lag. All dynamic variants outperform the fixed sparsity baseline (implied but not plotted).

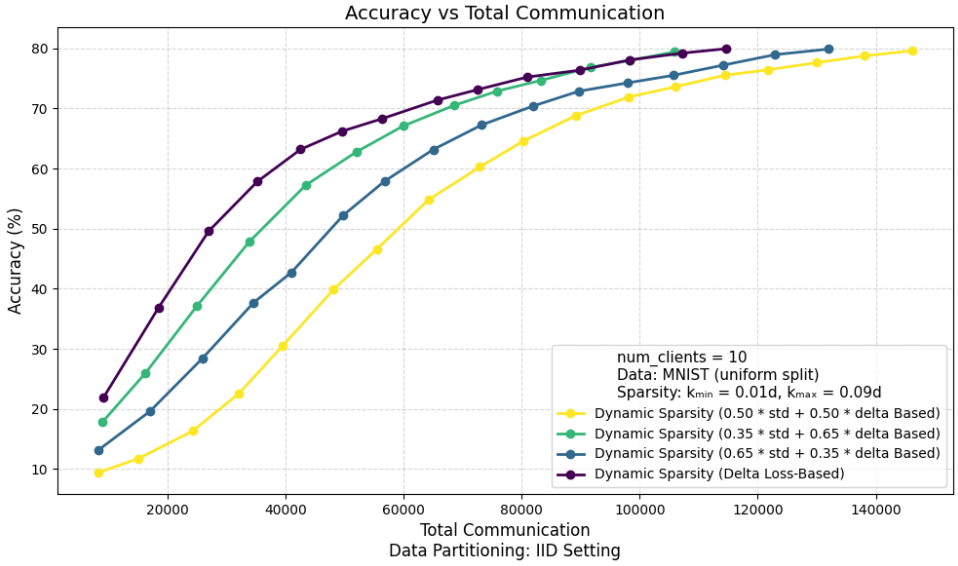

Figure 7.3: Hybrid Delta-Std strategies vs pure Delta Loss-Based sparsity. The **Delta Loss-Based** method achieves the fastest convergence, surpassing **70% accuracy** before competitors reach **60%**. Notably, the **0.35*std+0.65*delta** hybrid reaches **80% accuracy** faster than other hybrids.

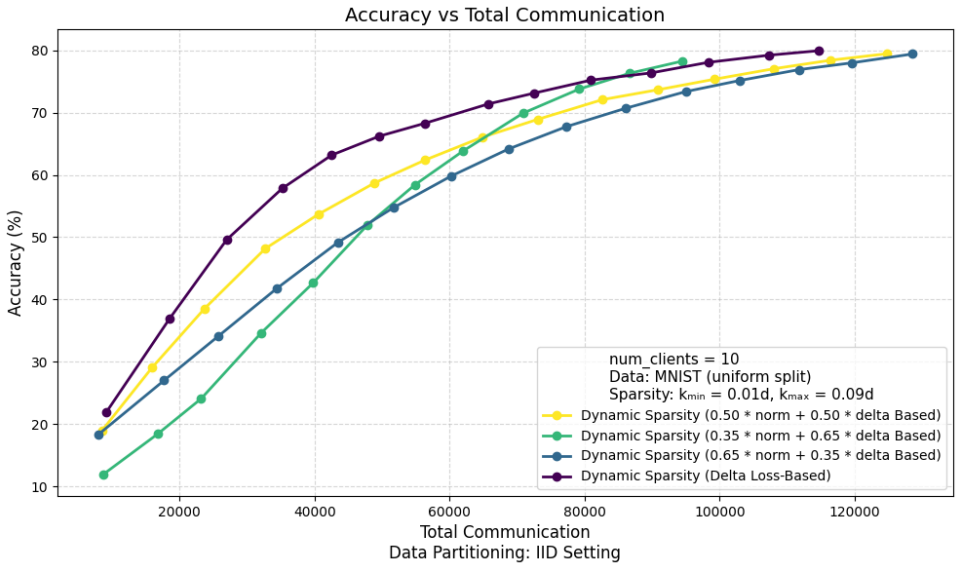

Figure 7.4: Hybrid Norm-Delta sparsity strategies in IID setting. While the **0.35*norm+0.65*delta** variant initially lags behind others, it demonstrates surprising late-stage efficiency, becoming the first hybrid to reach **80% accuracy**. The pure **Delta Loss-Based** method maintains overall dominance, crossing **60% accuracy** by **40k communication units**.

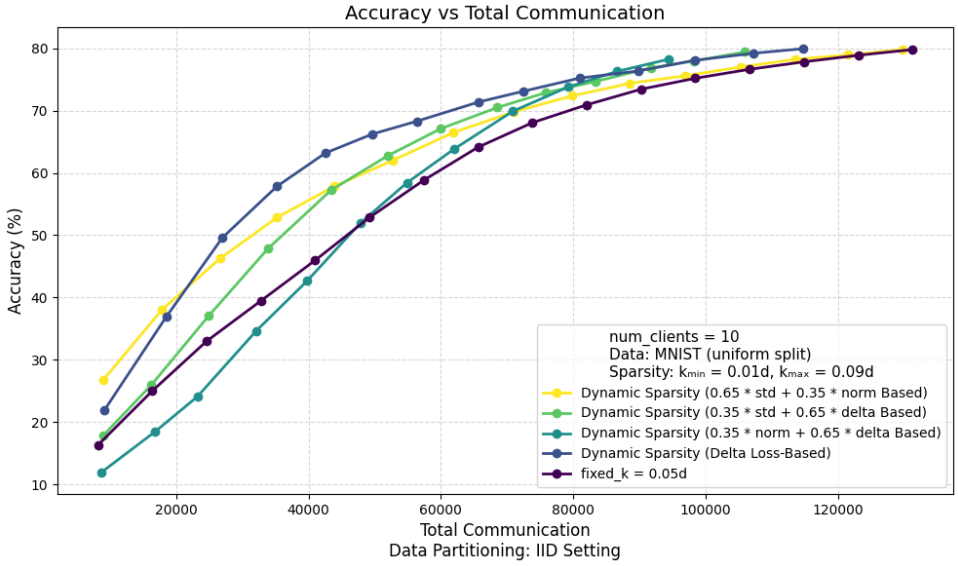

Figure 7.5: Comprehensive comparison of dynamic sparsity hybrids against fixed sparsity baseline. The **Delta Loss-Based** method maintains its dominance, reaching **60% accuracy** by **40k units** and outperforming all other approaches. Among hybrids, the **0.35*norm+0.65*delta** variant shows the most competitive performance.

7.3 Results and Analysis: NIID Setting

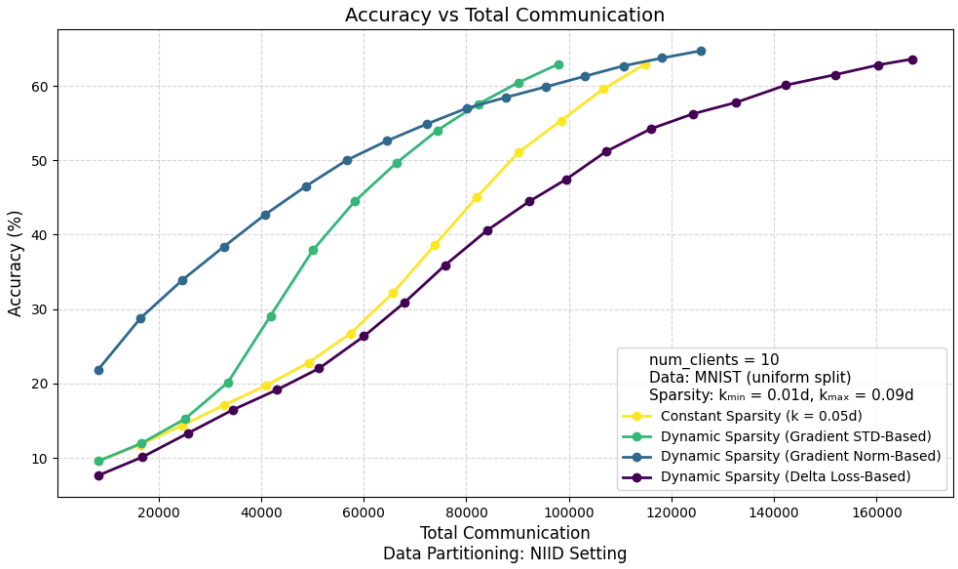

Figure 7.6: Key findings in heterogeneous data environments: (1) **Gradient STD-Based** sparsity emerges as the most effective approach, achieving **65% accuracy** faster (2) **delta-Based** strategy shows poor performance. (3) While **Norm-Based** methods show intermediate performance in later on.

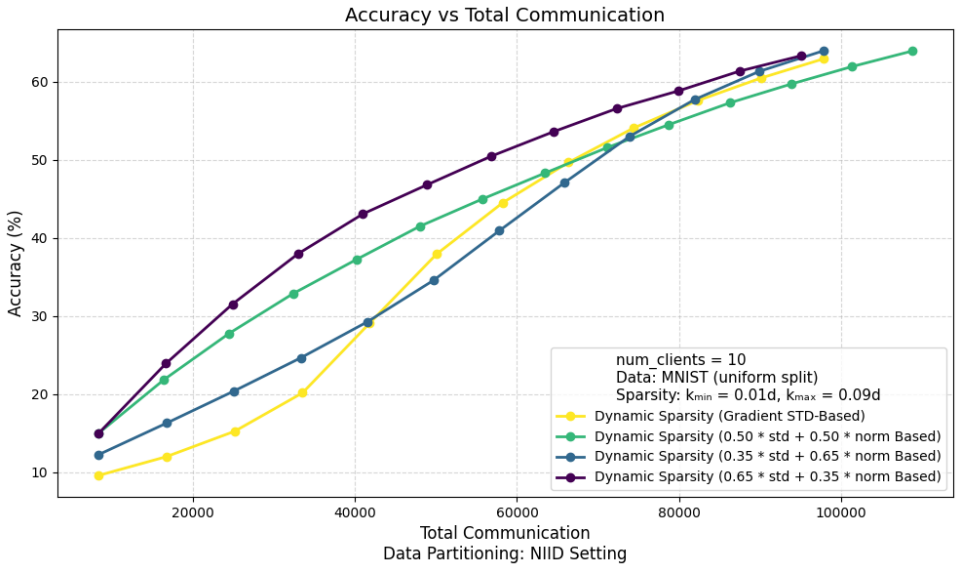

Figure 7.7: (1) Pure **STD-Based** sparsity achieves best performance (30% accuracy), maintaining consistent convergence. (2) Hybrid strategies demonstrate clear weighting tradeoffs - the **0.65*std+0.35*norm** variant performs closest to pure STD, while higher norm ratios degrade accuracy by 5-8%.

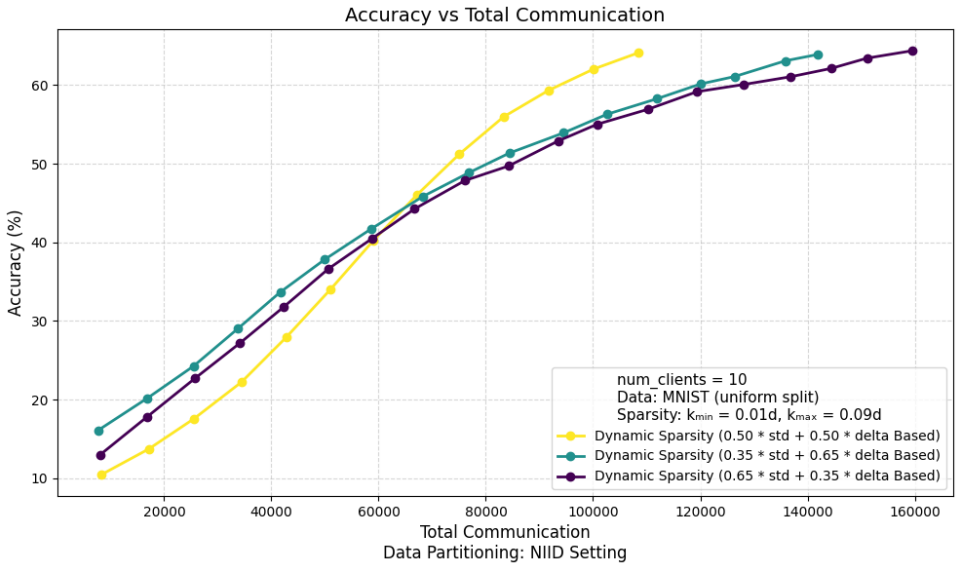

Figure 7.8: Hybrid STD-Delta evaluation reveals: (1) **0.50*std+0.50*delta** showing better late-stage convergence than other hybrids. (2) All other hybrids exhibit similar early performance (within 3% difference up to 60k units).

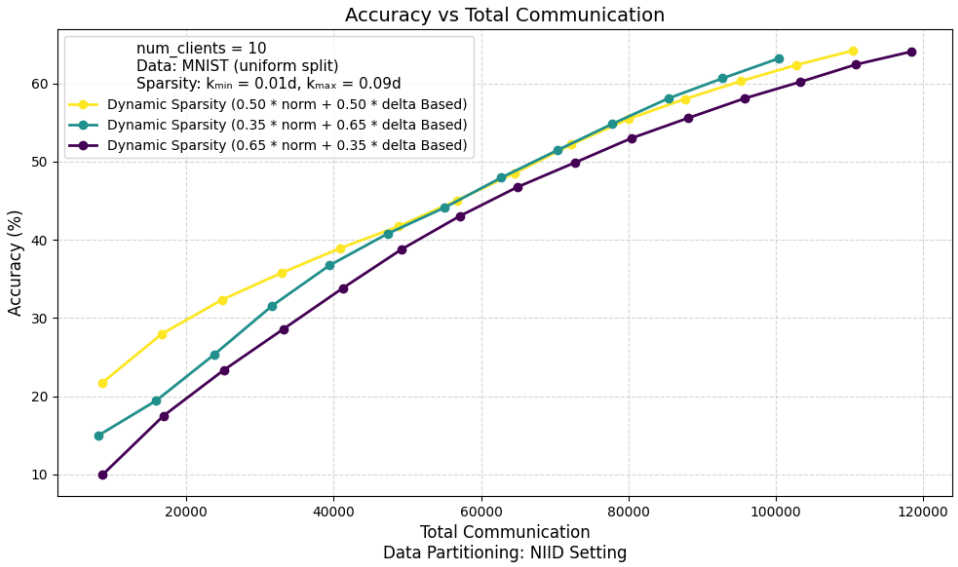

Figure 7.9: Norm-Delta hybrid analysis indicates: (1) **0.35*norm+0.65*delta** delivers strongest performance (35% peak accuracy), maintaining consistent growth throughout training. (2) The 50-50 hybrid shows intermediate results, while **0.65*norm+0.35*delta** underperforms by 6-8% accuracy after 60k units.

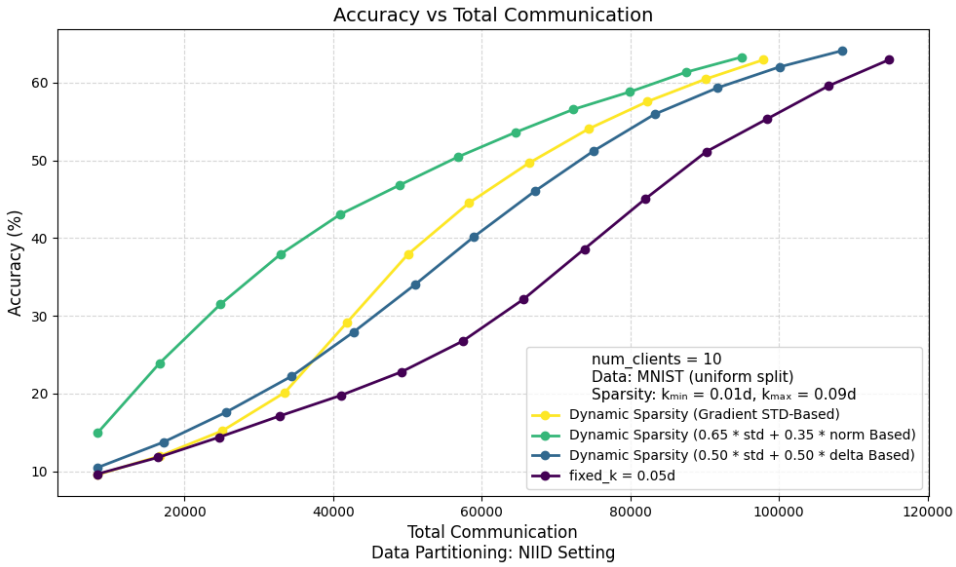

Figure 7.10: (1) The **0.65*std + 0.35*norm** hybrid demonstrates better stability than the 50-50 std-delta variant, particularly after 60k units. This clearly indicates that in the Non-IID setting, **gradient standard deviation** has a strong influence on effective dynamic sparsity selection.

Chapter 8

Conclusion

Conclusion

Dynamic sparsity demonstrates clear communication efficiency gains over constant sparsity, achieving a 31.4% reduction in communicated units for IID data (96k vs. 140k), and an 18.3% reduction for Non-IID data (98k vs. 120k). In addition to improved communication efficiency, the approach yields higher model accuracy, with absolute gains ranging from 5% to 15%.

Further analysis reveals that under IID (uniform) data distribution, the standard deviation of gradients contributes less to the sparsity decision, while the ℓ_2 norm and loss reduction metrics provide more reliable signals. In contrast, for Non-IID distributions, the standard deviation and the change in local loss serve, norm as stronger indicators for adaptive sparsity allocation.

Future Work: One direction is the use of reinforcement learning to dynamically tune the sparsity weighting parameters (α, β, γ) during training, allowing for real-time adaptation across rounds and clients. Another exciting avenue is the integration of differential privacy into the dynamic sparsity framework, enabling secure and communication-efficient federated learning with formal privacy guarantees.

Bibliography

[1] H. U. Sami and B. Güler, "Secure gradient aggregation with sparsification for resource-limited federated learning," *IEEE Transactions on Communications*, Aug. 2024.

[2] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Int. Conf. on Artificial Intelligence and Statistics (AISTATS)*, 2017.

[3] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, "Inverting gradients: How easy is it to break privacy in federated learning?" in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[4] 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 *Proc. IEEE Int. Conf. on Distributed Computing Systems (ICDCS)*, 2021.

[5] B. Zhao, P. Sun, T. Wang, and K. Jiang, "FedInv: Byzantine-robust federated learning by inversing local model updates," in *Proc. AAAI Conf. on Artificial Intelligence*, 2022.

[6] Q. Yu, S. Li, N. Raviv, S. M. M. Kalan, M. Soltanolkotabi, and S. A. Avestimehr, "Lagrange coded computing: Optimal design for resiliency, security, and privacy," in *Proc. Int. Conf. on Artificial Intelligence and Statistics (AISTATS)*, 2019.