Communication-Efficient State Synchronization for Stable Second-Order Federated Learning

Ahmed Hany[1], Karim Banawan[2,3], Nourhan Sakr[1], Karim Seddik[2], and Tamer ElBatt[1]

[1]Computer Science and Engineering Department, American University in Cairo, Egypt
[2]Electronics and Communications Engineering Department, American University in Cairo, Egypt
[3]Department of Electrical Engineering, Faculty of Engineering, Alexandria University, Egypt

***Abstract*—We consider the problem of designing a stable, fast-converging, and communication-efficient training procedure for federated learning (FL). First, we study Fed-Sophia, which is a second-order optimization method that promises to accelerate convergence by leveraging curvature information. Our empirical study reveals a critical instability under extreme non-IID datasets, which we term *cross-over divergence*. We identify the cause of the instability to the *mismatch* between the server and clients' optimizer states (i.e., momentum and Hessian estimates). To tackle this issue, we propose *Second-Order State Synchronization* (SOSS-FL), a communication-efficient protocol that exchanges optimizer states instead of model weights as in traditional federated Sophia. Thus, our proposed scheme offers convergence acceleration of second-order methods, eliminates the crossover divergence problem while maintaining communication costs similar to those of first-order methods. We conduct extensive experiments on non-IID Fashion-MNIST. Our results show that SOSS-FL achieves a 3% accuracy enhancement and an 11-fold speedup over FedAvg. When endowed with layer-wise quantization, our proposed approach maintains these gains while compressing the communication cost by $4.85\times$.**

***Index Terms*—Federated Learning, Second-Order Optimization, Sophia Optimizer, Non-IID Data, Communication Efficiency.**

I. Introduction

The rapid growth of intelligent devices at the network edge (e.g., IoT devices) poses significant challenges for traditional machine learning (ML) techniques, which rely on centralized data collection for training. This approach poses severe privacy risks in addition to incurring massive bandwidth costs of transmitting large datasets over wireless networks [1]–[3]. Federated Learning (FL) has emerged as a practical solution to these challenges. By enabling distributed clients to collaboratively train a global model without sharing raw data, FL ensures both privacy preservation and communication efficiency [4].

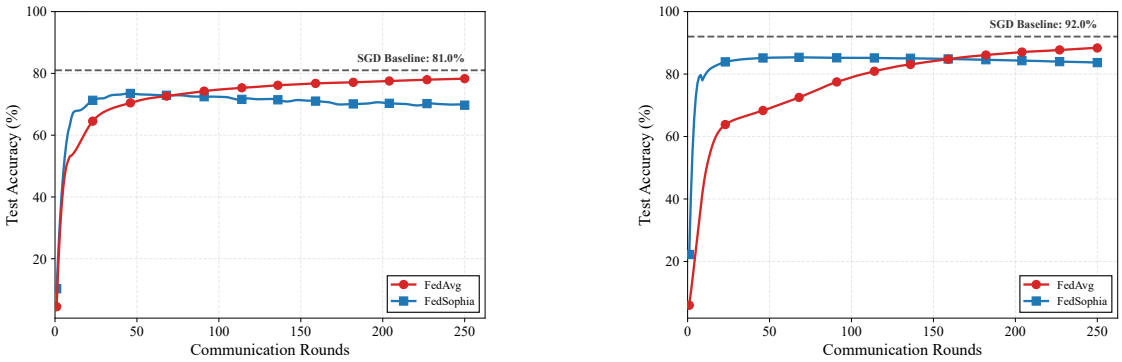
The objective of this work is to design a communication-efficient FL framework that exhibits accelerated convergence and stable training behavior. Aiming for convergence acceleration, we adopt second-order optimization techniques [7]–[9]. Specifically, we build on the Fed-Sophia framework introduced in [10] as a federated implementation of the Sophia optimizer [9]. Sophia is a second-order optimizer that employs a lightweight diagonal Hessian estimator, reducing both computation and memory requirements compared to classical second-order methods. This makes it attractive for edge deployment.

However, our experiments with traditional Fed-Sophia with extreme non-IID datasets (see Section V for the set-up) show a critical stability issue, which we term *cross-over divergence*. As shown in Fig. 1, Fed-Sophia initially accelerates training, outperforming first-order baselines, but often becomes unstable in later stages, eventually underperforming simpler methods. We identify the root cause of the instability issue to be an artifact of *state mismatch* between the server and clients. Specifically, in Fed-Sophia with heterogeneous datasets, when the server aggregates model updates without synchronizing the optimizer's states (e.g., momentum and curvature information), a *disagreement* emerges between the states at both server and client. This causes the global model trajectory to diverge from that implied by the clients' local states.

Prior work identified similar mismatches in first-order momentum-based methods [11]. However, state mismatch in second-order optimizers has not been systematically addressed. A common workaround in the literature is to restrict clients to a single local iteration per round [12]–[14]. This mitigates divergence but severely limits the computational efficiency and communication benefits. On the other hand, one may consider full synchronization by broadcasting all optimizer states (weights, momentum, and curvature) every round to ensure stability. This dramatically increases communication costs, which is prohibitive in wireless edge networks [15]. This raises an interesting question, namely, how to stabilize second-order federated optimization with multiple local steps without incurring the high communication cost of full state synchronization.

To address these challenges, we propose *Second-Order State Synchronization* (SOSS-FL), a communication-efficient protocol that resolves State Mismatch in Fed-Sophia. In SOSS-FL, the server provides clients with aggregated global momentum in addition to occasional global curvature information (every τ training rounds). This contrasts with traditional Fed-Sophia, which communicates the weights of the global model. By using these aggregated states and leveraging the last-stored global model (a.k.a., anchor model), clients can *reconstruct* the global model locally. SOSS-FL further reduces communication overhead by adopting layer-wise quantization. Consequently, the proposed approach mitigates cross-over divergence by resolving state mismatch between the server and clients, enjoys the promised second-order accelerated convergence, while reducing communication overhead.

We summarize our main contributions as follows:



(a) Fashion-MNIST (b) MNIST

Fig. 1. Illustration of Cross-Over Divergence. Test accuracy on Fashion-MNIST (left) and MNIST (right). The federated Sophia optimizer (Blue) initially accelerates training compared to FedAvg (Red) but becomes unstable in later stages, eventually underperforming FedAvg due to State Mismatch.

1) We empirically show that training instability (cross-over divergence) observed in Fed-Sophia [10] under non-IID data distributions is due to inconsistent optimizer states (i.e., momentum and Hessian) between server and clients.
2) We propose SOSS-FL, a second-order approach, which recovers the global model locally using the server's optimizer states without transmitting full model weights. This ensures stable and accelerated training.
3) To reduce the communication cost of SOSS-FL, we incorporate layer-wise quantization of the state vectors. Moreover, we limit the exchange of curvature information to once every τ rounds.
4) We conduct extensive experiments on the Fashion-MINIST dataset. Experimental results show the superiority of our proposed approach in terms of test accuracy, convergence speed, and stability.

Paper Organization: Section II surveys related work. Section III introduces the system model and problem formulation. Section IV presents the necessary preliminaries. Section V provides an empirical diagnosis of the cross-over divergence phenomenon. Section VI details the proposed SOSS-FL algorithm. Section VII reports the experimental evaluation, and Section VIII concludes the paper.

II. Related Work

FL has been extensively studied from the perspectives of optimization efficiency and communication reduction [1], [17]. First-order methods are widely used in FL due to their simplicity and low computational cost. The classical algorithm, FedAvg [1], reduces communication by allowing clients to perform J local updates before sending their model updates to the server. FedAvg suffers from *client drift* in non-IID settings. To address this issue, FedProx [11] introduces a proximal regularization term to constrain local updates toward the global model, while SCAFFOLD [5] employs control variates to correct the local gradient drift. Despite these enhancements, first-order methods suffer from slow training convergence as they only rely on gradient information. In bandwidth-limited networks, slow convergence implies excessive communication rounds, which significantly limit overall efficiency [6].

To accelerate convergence, recent works employ second-order optimization methods that leverage curvature information via the Hessian matrix, yielding faster convergence than first-order methods [7]–[9]. Algorithms such as FedNew [18] and GIANT [13] approximate the global Hessian using distributed clients. Nevertheless, these methods are expensive in terms of communication/computation costs of the Hessian matrix, especially for deep networks. Lightweight approaches, such as Fed-Sophia [10], reduce this cost by estimating only the diagonal of the Hessian. As we will see in Section V, second-order methods, particularly Fed-Sophia, suffer from training instability in extreme non-IID settings, which wastes their potential as a convergence acceleration method.

Furthermore, several works tackle the problem of reducing communication overhead in FL via state compression. Specifically, transmitted gradients or model updates are quantized or sparsified, as in QSGD [19]. While highly effective for first-order optimization, applying these techniques directly to second-order methods is challenging. Specifically, in second-order optimizers, the model update scales inversely with the local Hessian estimate. Thus, naive quantization of curvature can amplify Hessian estimation errors, leading to large variations in the effective step size and potentially causing divergence.

In the sequel, we show that SOSS-FL overcomes the aforementioned limitations by leveraging the structural properties of Sophia to enable accelerated training, avoiding training instability observed for non-IID settings, and facilitating high-fidelity quantization.

III. System Model and Problem Formulation

We consider a federated learning (FL) system comprising a central parameter server (PS) and N decentralized clients. The training process occurs over communication rounds indexed by $r = 0, 1, \ldots, R-1$. In each round, all clients perform J local update steps to optimize their specific objectives.

The k-th client aims to train a machine learning model $\boldsymbol{\theta} \in \mathbb{R}^d$ (e.g., neural network weights) using a private dataset $\mathcal{D}_k$. This dataset contains L_k labeled samples, denoted as $\mathcal{D}_k = \{(\mathbf{x}_{k,i}, y_{k,i})\}_{i=1}^{L_k}$, where $\mathbf{x}_{k,i}$ is the feature vector and $y_{k,i}$ is the label. We explicitly address the *Non-IID setting*, where data

distributions vary across clients. The k-th client minimizes the local loss function $F_k(\boldsymbol{\theta})$, defined as:

$$F_k(\boldsymbol{\theta}) = \frac{1}{L_k} \sum_{i=1}^{L_k} \ell(\boldsymbol{\theta}; (\mathbf{x}_{k,i}, y_{k,i})) \tag{1}$$

where $\ell(\cdot)$ is the sample-wise loss function.

The global objective of the FL system is to collaboratively minimize the weighted average loss $f(\boldsymbol{\theta})$:

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^d} f(\boldsymbol{\theta}) \triangleq \sum_{k=1}^{N} p_k F_k(\boldsymbol{\theta}) \tag{2}$$

where $p_k \geq 0$ is the k-th client weight, satisfying $\sum_{k=1}^{N} p_k = 1$. In this work, we adopt uniform weights[1], i.e., $p_k = \frac{1}{N}$.

We assume a synchronous communication protocol with full client participation. Specifically, at the rth round, for all $k = 1, \cdots, N$, the k-th client sends a *client update*, denoted by $U_k^{(r)}$, to the PS. $U_k^{(r)}$ contains local optimization information derived from the J local steps. The PS aggregates these updates and broadcasts a *server update*, denoted by $U_s^{(r)}$, back to the clients.

The objective of this work is to design communication-efficient schemes for the client update $U_k^{(r)}$ and server update $U_s^{(r)}$ that minimize the weighted average loss $f(\boldsymbol{\theta})$.

IV. Preliminaries

In this section, we provide a brief overview of the original Sophia optimizer [9] and its federated version, Fed-Sophia [10]. Fed-Sophia serves as a benchmark to our proposed work.

A. The Sophia Optimizer

The Second-order Clipped Stochastic Optimization (Sophia) is a light-weight second-order optimizer [9], that utilizes the Exponential Moving Average (EMA) of the *diagonal* Hessian estimates. Specifically, at the jth step, centralized Sophia maintains two internal states, the gradient EMA ($\mathbf{m}_j$) and the diagonal Hessian EMA ($\mathbf{h}_j$). The update rule is given by[2]:

$$\boldsymbol{\theta}_{j+1} = \boldsymbol{\theta}_j - \eta \cdot \text{clip}\left(\frac{\mathbf{m}_j}{\mathbf{h}_j + \epsilon}, \rho\right) \tag{3}$$

where $\text{clip}(\cdot, \rho)$ bounds the update magnitude to ensure stability to a clipping threshold ρ, η is the learning rate, and ϵ is a damping factor. The estimators update as:

$$\mathbf{m}_j = \beta_1 \mathbf{m}_{j-1} + (1 - \beta_1)\mathbf{g}_j \tag{4}$$

$$\mathbf{h}_j = \beta_2 \mathbf{h}_{j-1} + (1 - \beta_2)\hat{\mathbf{h}}_j \tag{5}$$

where $\beta_1, \beta_2 \in [0, 1)$ are decay rates for the gradient and Hessian, respectively. $\mathbf{g}_j$ denotes the stochastic gradient, and $\hat{\mathbf{h}}_j$ is the instantaneous diagonal Hessian estimate. In this work, we use the Gauss-Newton-Bartlett (GNB) estimator [9]:

$$\hat{\mathbf{h}}_j = \hat{\mathbf{g}}_j \odot \hat{\mathbf{g}}_j, \tag{6}$$

where $\odot$ denotes the element-wise product, and $\hat{\mathbf{g}}_j$ is the sampled gradient estimate.

[1]Uniform weights prevent clients with large datasets from biasing the global curvature estimates of the second-order optimization.

[2]Vector division denotes element-by-element division throughout this paper.

TABLE I
Experimental Hyperparameters

Parameter	Value
Dataset	Fashion-MNIST
Model Architecture	MLP ($784 \to 100 \to 10$)
Number of Clients (N)	32
Communication Rounds (R)	250
Local Epochs (J)	10
Batch Size (B)	512
Learning Rate (η)	0.003
Hessian Sync Interval (τ)	10 rounds
Sophia clipping (ρ)	5
Sophia β_1, β_2	0.965, 0.95
Damping factor (ϵ)	1e-15

B. Direct Federated Implementation of Sophia (Fed-Sophia)

Next, we consider the direct decentralized implementation of Sophia, namely, Fed-Sophia [10], as our main benchmark. In Fed-Sophia, at the start of the rth round, the PS broadcasts the global model $\boldsymbol{\Theta}^{(r)}$ as its server update, i.e., $U_s^{(r)} = \boldsymbol{\Theta}^{(r)}$.

The k-th client maintains local optimizer states for the gradient EMA (a.k.a., momentum) and Hessian EMA. The k-th client initializes its local model $\boldsymbol{\theta}_{k,0}^{(r)} = \boldsymbol{\Theta}^{(r)}$ and performs J local updates. For the local step $j = 0, \ldots, J-1$, the client updates $\mathbf{m}_{k,j}^{(r)}$ using the local stochastic gradient. The Hessian EMA $\mathbf{h}_k^{(r)}$ is periodically updated every τ rounds to reduce computational overhead. The local model parameter update is:

$$\boldsymbol{\theta}_{k,j+1}^{(r)} = \boldsymbol{\theta}_{k,j}^{(r)} - \eta \cdot \text{clip}\left(\frac{\mathbf{m}_{k,j}^{(r)}}{\mathbf{h}_{k,j}^{(r)} + \epsilon}, \rho\right) \tag{7}$$

After J steps, The k-th client sends a client update in the form of the local model, i.e., $U_k^{(r)} = \boldsymbol{\theta}_{k,J}^{(r)}$. The server aggregates the locally trained models to form the next global state:

$$\boldsymbol{\Theta}^{(r+1)} = \frac{1}{N} \sum_{k=1}^{N} \boldsymbol{\theta}_{k,J}^{(r)} \tag{8}$$

Crucially, in Fed-Sophia, the optimizer states $\mathbf{m}_k$ and $\mathbf{h}_k$ remain local and not shared with the PS, and vice versa. This creates a *State mismatch* between clients, as the aggregated global model $\boldsymbol{\Theta}^{(r+1)}$ becomes misaligned with the local curvature states $\mathbf{h}_k^{(r+1)}$ and local momentum $\mathbf{m}_k^{(r+1)}$ at the start of the subsequent round. This mismatch leads to poor convergence in heterogeneous (Non-IID) datasets as we will investigate next.

V. Motivation: Empirical Diagnosis of Cross-Over Divergence

We motivate our proposed approach by presenting a concrete training scenario in which Fed-Sophia fails to converge stably, particularly when clients perform multiple local update steps ($J > 1$), and non-IID distributions. We term this phenomenon the *cross-over divergence*. We conduct a diagnostic analysis to identify the root cause of this failure. Our empirical study reveals that the instability stems from the *State Mismatch*, where clients apply private, decoupled optimizer states ($\mathbf{m}_k, \mathbf{h}_k$) to a newly aggregated global model with which they are misaligned. Thus, although curvature-aware updates offer acceleration benefits [8], [9], these incoherent updates ultimately destabilize training.

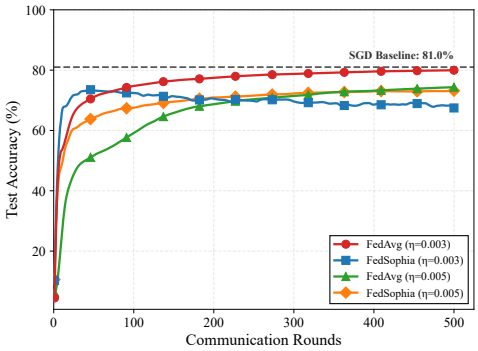


Fig. 2. Effect of Learning Rate (η). Reducing learning rate delays the onset of instability but does not prevent the eventual cross-over.

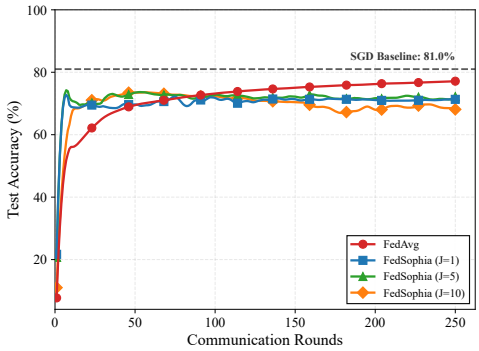


Fig. 3. Effect of Local Steps (J). Increasing the number of local update steps J significantly accelerates the instability.

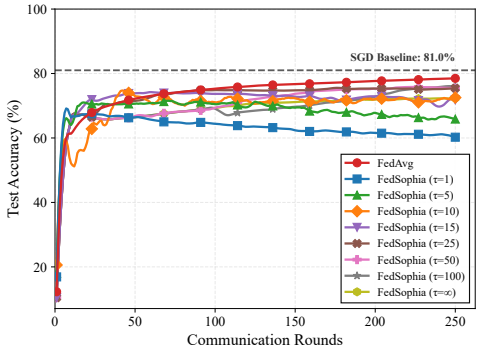


Fig. 4. Effect of Hessian Update Frequency (τ). Increasing the frequency of Hessian synchronization alone is insufficient to prevent Cross-Over Divergence.

A. Experimental Setup for Diagnosis

To show the effect of extreme heterogeneity in the clients' datasets, we use a restricted class partition as a model for non-IID datasets, as in [1]. Let $\mathcal{C} = \{1, \ldots, C\}$ denote the set of all classes in the dataset. The k-th client is assigned a subset of classes $\mathcal{C}_k \subset \mathcal{C}$ such that $|\mathcal{C}_k| = S$, with $S < C$.

Specifically, we train a Multilayer Perceptron (MLP) for 250 communication rounds using the hyperparameters detailed in Table I. We use the Fashion-MNIST dataset, denoted by $\mathcal{D}$, in our experiments with $C = 10$ and $S = 3$. Consistent with the system model, the local dataset $\mathcal{D}_k$ is composed of L_k samples drawn from $\mathcal{D}$ that match the client's assigned classes:

$$\mathcal{D}_k = \{(\boldsymbol{x}_{k,i}, y_{k,i}) \in \mathcal{D} : y_{k,i} \in \mathcal{C}_k\}_{i=1}^{L_k} \tag{9}$$

B. The Cross-Over Divergence Phenomenon

Fig. 1 shows the learning curve for FedAvg and Fed-Sophia corresponding to the mentioned set-up. In the early phase (rounds 0-30), Fed-Sophia exhibits the expected second-order acceleration, reaching 75% accuracy faster than FedAvg. Unfortunately, at round 40, Fed-Sophia test accuracy falls below the FedAvg curve. By round 250, the slower FedAvg algorithm achieves $\sim 80\%$ accuracy, while Fed-Sophia degrades to $\sim 70\%$. This cross-over divergence indicates that while local curvature estimates provide an initial acceleration, they diverge in later stages of training, resulting in structural instability.

C. Sensitivity Analysis: Hyperparameter Impact

To determine whether the observed degradation was an artifact of suboptimal hyperparameter tuning, we conducted a series of ablation studies. We independently analyze each hyperparameter to isolate its effect on the said divergence:

- *Learning Rate (η):* Fig. 2 shows that although reducing the learning rate from 0.003 to 0.0005 delays the onset of instability, it does not prevent the eventual cross-over. Fed-Sophia continued to degrade below the FedAvg, falling from a peak of $\sim 75\%$ to $\sim 70\%$. This confirms that the instability is not a result of poorly tuned global steps.
- *Local Steps (J):* Fig. 3 illustrates that increasing the number of local updates significantly accelerates the divergence. Even with $J = 1$, the algorithm struggles to maintain stability, exhibiting both suboptimal convergence and high communication cost. The rapid degradation observed at $J = 10$ further indicates that local optimizer states become stale and misleading when decoupled from the global trajectory.
- *Hessian Update Frequency (τ):* Fig. 4 shows that even for $\tau = 1$ (updating the Hessian estimate each round), the cross-over divergence persists. This indicates that synchronizing the Hessian more frequently is insufficient to restore stability if the local and global states are misaligned.

D. Analysis of Gradient Drift and State Mismatch

Next, we examine whether the instability originated from gradient misalignment (i.e., drift) across clients by integrating SCAFFOLD drift correction [5], and FedProx regularization [11] into Fed-Sophia. Fig. 5 shows that applying either method fails to resolve the cross-over divergence as both settle at test accuracy $\sim 70\%$. Thus, while drift correction methods successfully correct the raw gradient vector $\mathbf{g}$, they do not address the divergence in the optimizer's internal states $(\mathbf{m}_k, \mathbf{h}_k)$.

Consequently, we identify the cause of the observed instability as *state mismatch*, i.e., the desynchronization of both the accumulated momentum and the EMA Hessian diagonal, which produces incoherent update steps across heterogeneous clients, and creates instability that first-order corrections cannot resolve.

VI. Proposed Algorithm: SOSS-FL

A straightforward solution for the state-mismatch problem in Section V is to construct the server update to be $U_s^{(r)} = (\boldsymbol{\Theta}, \mathbf{m}_s, \mathbf{h}_s)$, and similarly the client update to be $U_k^{(r)} =$

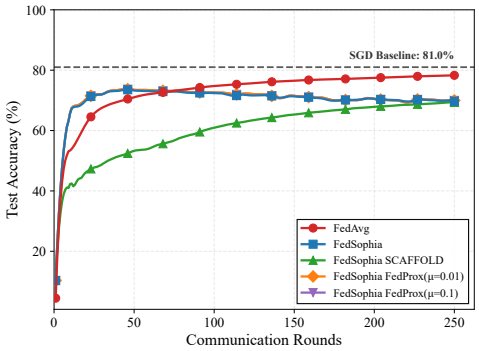


Fig. 5. Failure of First-Order Drift Correction. Standard corrections (SCAFFOLD, FedProx) fail to stabilize Fed-Sophia, confirming that the issue requires synchronizing the second-order state.

$(\boldsymbol{\theta}_k, \mathbf{m}_k, \mathbf{h}_k)$. This re-synchronizes the complete training state of the clients and restores learning stability. However, this approach incurs a communication cost of $6d$ elements per round (sending 3 vectors in both uplink and downlink), compared to only $2d$ element in standard Fed-Sophia and FedAvg.

In this work, we propose *Second-Order State Synchronization* (SOSS-FL), a protocol designed to eliminate the cross-over divergence observed in Fed-Sophia without compromising communication efficiency. The method enforces state coherence by having all clients maintain a locally stored *anchor model*, $\boldsymbol{\Theta}_{\text{anchor}} \equiv \boldsymbol{\Theta}^{(r-1)}$, which acts as a reference for the global optimization trajectory.

In this protocol, the PS broadcasts a server update $U_s^{(r)}$ composed of the aggregated optimizer states: the momentum $\mathbf{m}$ and the Hessian estimate $\mathbf{h}$. Leveraging the deterministic update rule of the optimizer, clients exactly SOSS-FL the updated global model $\boldsymbol{\Theta}^{(r)}$ locally from the anchor model and state vectors. To optimize bandwidth utilization, we introduce a multi-rate synchronization schedule based on the temporal stability of the Hessian. While the momentum $\mathbf{m}$ is synchronized at every round to track rapid gradient changes, the curvature information $\mathbf{h}$ is updated with period τ. This yields a bidirectional communication complexity of $\approx 2\left(1+\frac{1}{\tau}\right)d$ per round, which asymptotically approaches the cost of first-order methods while achieving stable second-order convergence.

A. The Reconstruction Principle

SOSS-FL addresses the state mismatch issue by enforcing state-level synchronization (i.e., $(\mathbf{m}, \mathbf{h})$ level), rather than parameter-level synchronization (i.e., $\boldsymbol{\Theta}$ level).

Initially ($r = 0$), the server broadcasts the model parameters $\boldsymbol{\Theta}^{(0)}$, which all clients store as the initial anchor $\boldsymbol{\Theta}_{\text{anchor}}$. For the rth round, $r \geq 1$, clients maintain the anchor model $\boldsymbol{\Theta}_{\text{anchor}} = \boldsymbol{\Theta}^{(r-1)}$ from the previous iteration. The server then broadcasts the update packet $U_s^{(r)} = (\mathbf{m}_s^{(r)}, \mathbf{h}_s^{(r)})$, where $\mathbf{m}_s^{(r)}$ and $\mathbf{h}_s^{(r)}$ correspond to the uniform aggregation of the momentum and Hessian at the server, respectively.

The k-th client, $k = 1, \ldots, N$, initializes $\mathbf{m}_{k,0}^{(r)} = \mathbf{m}_s^{(r)}$ and updates the momentum estimate at the jth local step as:

$$\mathbf{m}_{k,j}^{(r)} = \beta_1 \mathbf{m}_{k,j-1}^{(r)} + (1-\beta_1)\mathbf{g}_{k,j}^{(r)}, \quad j = 1, \ldots, J \tag{10}$$

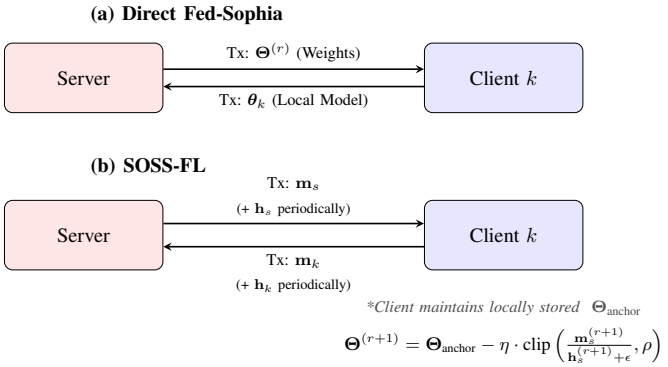


Fig. 6. Communication Comparison: (a) Direct Federated Implementation of Sophia (Fed-Sophia) transmits full model weights (downlink) and trained local models (uplink). (b) SOSS-FL transmits only optimizer states (momentum $\mathbf{m}$, plus Hessian $\mathbf{h}$ periodically). The client reconstructs the global update locally using the shared anchor model.

The Hessian estimate updates every τ rounds as:

$$\mathbf{h}_{k,j}^{(r)} = \beta_2 \mathbf{h}_{k,j-1}^{(r)} + (1-\beta_2)\hat{\mathbf{h}}_{k,j}^{(r)}, \quad j = 1, \ldots, J \tag{11}$$

Note that the server's Hessian estimate is transmitted every τ communication rounds, hence, the initial local state is,

$$\mathbf{h}_{k,0}^{(r)} = \begin{cases} \mathbf{h}_s^{(r)}, & \text{if } r \bmod \tau = 0, \\ \mathbf{h}_s^{(\lfloor \frac{r}{\tau} \rfloor \tau)}, & \text{otherwise.} \end{cases} \tag{12}$$

where $\lfloor \cdot \rfloor$ denotes the floor function.

Thus, all local optimization trajectories begin from identical global states, eliminating state mismatch at its source. After completing J local steps, clients transmit their final optimizer states $U_k^{(r)}$ to the server. Consistent with the downlink schedule, the uplink transmission is also periodic:

$$U_k^{(r)} = \begin{cases} (\mathbf{m}_{k,J}^{(r)}, \mathbf{h}_{k,J}^{(r)}), & \text{if } r \bmod \tau = 0, \\ (\mathbf{m}_{k,J}^{(r)}), & \text{otherwise.} \end{cases} \tag{13}$$

The server aggregates these states to form the global momentum $\mathbf{m}_s^{(r+1)}$ and, when applicable, the global Hessian estimate $\mathbf{h}_s^{(r+1)}$ as follows:

$$\mathbf{m}_s^{(r+1)} = \frac{1}{N}\sum_{k=1}^{N} \mathbf{m}_{k,J}^{(r)}, \quad \forall r \tag{14}$$

$$\mathbf{h}_s^{(r+1)} = \frac{1}{N}\sum_{k=1}^{N} \mathbf{h}_{k,J}^{(r)}, \quad \text{if } r \bmod \tau = 0 \tag{15}$$

Upon receiving $(\mathbf{m}_s^{(r+1)}, \mathbf{h}_s^{(r+1)})$, each client reconstructs the updated global model locally using its anchor as:

$$\boldsymbol{\Theta}^{(r+1)} = \boldsymbol{\Theta}_{\text{anchor}} - \eta \cdot \text{clip}\left(\frac{\mathbf{m}_s^{(r+1)}}{\mathbf{h}_s^{(r+1)} + \epsilon}, \rho\right) \tag{16}$$

This reconstruction exactly matches the Fed-Sophia update without explicit model communication. Following reconstruction, each client updates its anchor by setting $\boldsymbol{\Theta}_{\text{anchor}} \leftarrow \boldsymbol{\Theta}^{(r+1)}$ and overwrites its local optimizer states with the synchronized global values. Consequently, every new round of local training begins from a fully aligned optimization trajectory.

B. Layer-Wise Stochastic Quantization

To further reduce communication overhead, SOSS-FL incorporates layer-wise stochastic quantization. Here, a layer refers to a distinct parameter block (e.g., a neural network layer) in the network. Since the optimizer states ($\mathbf{m}$,$\mathbf{h}$) preserve the same structural dimensions as the model, we partition these state vectors into L layers. The ℓth layer consists of d_ℓ elements from the full ($\mathbf{m}$,$\mathbf{h}$) state. We denote the ℓth partition of the moment vector as $\mathbf{m}_\ell$ and $\mathbf{h}_\ell$ for the Hessian estimate partition. We apply the quantization for $\mathbf{m}_\ell$, and $\mathbf{h}_\ell$ independently for $\ell = 1, \cdots, L$. This approach minimizes quantization error by accommodating the scale heterogeneity often present between different layers [19].

Denote $\mathbf{v} \in \mathbb{R}^{d_l}$ to be either $\mathbf{m}_\ell$ or $\mathbf{h}_\ell$ in the ℓth layer, then the B-bit quantized representation is defined as[3]:

$$Q(v_i) = \|\mathbf{v}\|_\infty \cdot \text{sign}(v_i) \cdot \frac{\lfloor |v_i|/\|\mathbf{v}\|_\infty \cdot (2^{B-1}-1) \rfloor}{2^{B-1}-1} \quad (17)$$

This quantization scheme is applied symmetrically to both uplink (client-to-server) and downlink (server-to-client) communication, ensuring consistent bandwidth reduction throughout the federated optimization process.

C. Algorithm Description

We provide a detailed pseudocode for the proposed SOSS-FL. The protocol is coordinated through two main procedures: The Server-Side procedure (Algorithm 1) and the Client-Side procedure (Algorithm 2).

1) Server-Side Procedure: For $r \geq 1$, the server neither maintains or transmits the global model weights $\mathbf{\Theta}$. Instead, it aggregates the momentum and, EMA of the diagonal Hessian estimates from all clients (if $\frac{r}{\tau} \in \mathbb{N}$). The server broadcasts the aggregated states to all clients to drive local reconstruction.

Algorithm 1 SOSS-FL (Server Side)

Initialize: Momentum $\mathbf{m}_s^{(0)} \leftarrow \mathbf{0}$, Hessian $\mathbf{h}_s^{(0)} \leftarrow \mathbf{0}$, Model $\mathbf{\Theta}^{(0)}$.
Initialization Broadcast: Send $\mathbf{\Theta}^{(0)}$ to all clients (to set $\mathbf{\Theta}_{\text{anchor}}$).
for round $r = 0, \ldots, R-1$ **do**
  Broadcast Update:
  if $r = 0$ or $r \bmod \tau = 1$ **then**
    Send $U_s^{(r)} \leftarrow \left\{\mathbf{m}_s^{(r)}, \mathbf{h}_s^{(r)}\right\}$ to all clients.
  else
    Send $U_s^{(r)} \leftarrow \left\{\mathbf{m}_s^{(r)}\right\}$ to all clients. *// Hessian is skipped*
  end if
  Receive: Optimizer states from clients $k = 1 \ldots N$:
    Receive $\mathbf{m}_{k,J}^{(r)}$ (Always).
    Receive $\mathbf{h}_{k,J}^{(r)}$ (Only if $r \bmod \tau = 0$).
  Aggregation :
  $\mathbf{m}_s^{(r+1)} \leftarrow \frac{1}{N}\sum_{k=1}^N \mathbf{m}_{k,J}^{(r)}$
  if $r \bmod \tau = 0$ **then**
    $\mathbf{h}_s^{(r+1)} \leftarrow \frac{1}{N}\sum_{k=1}^N \mathbf{h}_{k,J}^{(r)}$
  end if
end for

[3]The communication cost of the quantization metadata consists of transmitting two full-precision scalars (e.g., scale and offset) per layer, resulting in an overhead of $2 \times 32 \times L$ bits. Since the model dimension $d \gg L$, this overhead is negligible compared to the total communication cost per round.

2) Client-Side Procedure: Algorithm 2 shows the rth round of the procedure, which contains two phases. In Phase 1 (Reconstruction), the client uses the received global optimizer states to update its local anchor model. In Phase 2 (Local Computation), the client performs standard local training. The client estimates curvature via GNB [9] only when $\frac{r}{\tau} \in \mathbb{N}$. The client sends the local optimizer state, i.e., the moement, and the EMA of Hessian (if possible) after J local steps.

Algorithm 2 SOSS-FL (Client Side, rth Round)

Persistent State: Anchor $\mathbf{\Theta}_{\text{anchor}}$, $\mathbf{h}_s$. *// Maintained across rounds*
Receive: Update $U_s^{(r)}$ from Server.
// — PHASE 1: RECONSTRUCTION & INITIALIZATION —
Unpack $\mathbf{m}_s$ from $U_s^{(r)}$.
$\mathbf{m}_k \leftarrow \mathbf{m}_s$ *// Reset local momentum to server state*
if $U_s^{(r)}$ contains $\mathbf{h}_s$ **then**
  $\mathbf{h}_k \leftarrow \mathbf{h}_s$ *// Sync local Hessian*
end if
Reconstruct Global Model:
$\mathbf{\Theta}^{(r)} \leftarrow \mathbf{\Theta}_{\text{anchor}} - \eta \cdot \text{clip}\left(\frac{\mathbf{m}_s}{\mathbf{h}_k+\epsilon}, \rho\right)$
$\mathbf{\Theta}_{\text{anchor}} \leftarrow \mathbf{\Theta}^{(r)}$ *// Update Anchor for next round*
$\boldsymbol{\theta}_k \leftarrow \mathbf{\Theta}^{(r)}$ *// Initialize local model*
// — PHASE 2: LOCAL COMPUTATION —
for local epoch $j = 0, \ldots, J-1$ **do**
  for batch $(\boldsymbol{x}, y)$ in $\mathcal{D}_k$ **do**
    $\mathbf{g} \leftarrow \nabla \ell(\boldsymbol{\theta}_k; \boldsymbol{x}, y)$
    $\mathbf{m}_k \leftarrow \beta_1 \mathbf{m}_k + (1-\beta_1)\mathbf{g}$ *// Update Momentum*
    Estimate Curvature:
    if $r \bmod \tau = 0$ **then**
      $\hat{\mathbf{h}} \leftarrow$ GNB_Estimator($\boldsymbol{\theta}_k$) [9]
      $\mathbf{h}_k \leftarrow \beta_2 \mathbf{h}_k + (1-\beta_2)\hat{\mathbf{h}}$ *// Update Hessian*
    end if
    Update Weights:
    $\boldsymbol{\theta}_k \leftarrow \boldsymbol{\theta}_k - \eta \cdot \text{clip}\left(\frac{\mathbf{m}_k}{\mathbf{h}_k+\epsilon}, \rho\right)$
  end for
end for
Uplink Transmission:
if $r \bmod \tau = 0$ **then**
  Return: $U_k^{(r)} = \{\mathbf{m}_k, \mathbf{h}_k\}$
else
  Return: $U_k^{(r)} = \{\mathbf{m}_k\}$
end if

VII. Experimental Evaluation

In this section, We empirically evaluate the efficiency and stability of SOSS-FL on the Fashion-MNIST dataset. Unless otherwise stated, the experimental set-up follows Section V. Experiments were implemented in PyTorch using standard deep learning libraries. All results are averaged over at least three randomized seeds to ensure consistency.

A. Experiment Benchmarks and Metrics

To show the efficacy of our proposed approach, we compare SOSS-FL against the following benchmarks: 1) *FedAvg:* Standard first-order SGD, 2) *Original Fed-Sophia* [10], which only exchanges the model weights (see Section IV), 3) *Fed-Sophia (3-Matrix)*, which exchanges the full optimizer state, i.e., model weights, momentum, and EMA of the Hessian estimate, and 4) *SGD centralized* baseline, which constructs an upper bound for the model performance without effects of federation. Metrics

TABLE II
COMMUNICATION COST PER ROUND

Algorithm	**Update Payload** (per round)	**Communication Cost** (*2 for Uplink+Downlink)	**Cost Evaluation** ($B = 32, \tau = 10$)
FedAvg	$\boldsymbol{\Theta}$	$2Bd$	$64d$
Fed-Sophia (3-Mat)	$\{\boldsymbol{\Theta}, \mathbf{m}, \mathbf{h}\}$	$2Bd\,(2 + \frac{1}{\tau})$	$134.4d$
Proposed	$\{\mathbf{m}, \mathbf{h}\}$	$2Bd\,(1 + \frac{1}{\tau})$	$70.4d$

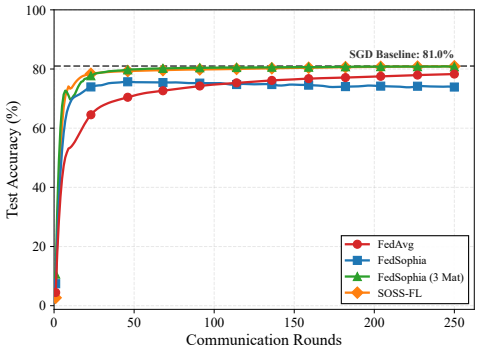


Fig. 7. Test accuracy over communication rounds. SOSS-FL (Orange) overlaps with the 3-Matrix baseline (Green), significantly outperforming Fed-Sophia (Blue) and the slower FedAvg (Red).

of interest include model test accuracy, convergence speed, and the per-round communication cost.

B. Communication Efficiency Analysis

Table II shows the per-round communication cost of the different protocols in terms of the number of bits B, model dimension d, and synchronization interval τ. We explicitly evaluate the communication cost for the case of $B = 32$, and $\tau = 10$. The 3-Matrix Fed-Sophia transmits weights and momentum every round, with Hessian updates every τ rounds. In contrast, SOSS-FL transmits only momentum at each round and the Hessian periodically. The proposed method reduces communication overhead by nearly 48% compared to the 3-Matrix baseline, achieving a cost comparable to FedAvg while preserving second-order convergence.

C. Test Accuracy Results

Fig. 7 shows the test accuracy versus communication rounds. Fig. 7 shows that SOSS-FL matches the test accuracy of 3-Matrix Fed-Sophia (Green), and the centralized SGD at $\sim 81\%$ with more communication-efficient updates.

In addition, SOSS-FL does not exhibit the cross-over divergence of the direct Fed-Sophia (Blue). The original Fed-Sophia stagnates at $\approx 75\%$ and falls to $\sim 70\%$ due to state mismatch in contrast to our proposed approach.

Moreover, Fig. 7 confirms the superiority of SOSS-FL over FedAvg in terms of test accuracy and convergence speed. Peak accuracy ($\approx$ 81%) of SOSS-FL surpasses FedAvg ($\approx$ 78%, Red curve). SOSS-FL reaches the target accuracy significantly faster, leveraging second-order information to accelerate training compared to the slower first-order updates of FedAvg.

D. Convergence Speed

Table III summarizes the performance of all benchmarks. Focusing on the convergence speed, we quantify the *convergence rounds* as the number of rounds needed to reach the peak accuracy of FedAvg (which is 78%).

TABLE III
PERFORMANCE SUMMARY (FASHION-MNIST)

Algorithm	**Peak Accuracy**	**Rounds to 78%**	**Cost**	**Comment**
FedAvg	78.5%	200	1.0×	Slow
Fed-Sophia (Orig)	75.4%	— (Diverged)	1.0×	Unstable
Fed-Sophia (3-Mat)	81.0%	18	2.1×	Expensive
Proposed	**81.1%**	**18**	**1.1×**	**Superior**

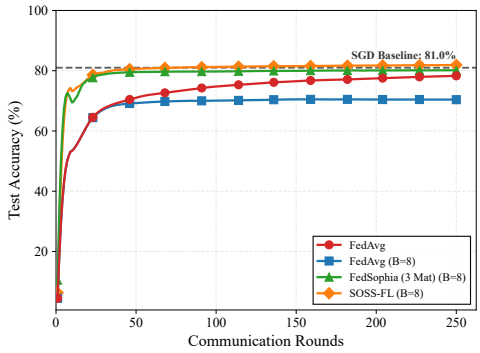


Fig. 8. 8-bit quantization robustness. Quantized FedAvg (Blue) collapses, while SOSS-FL (Orange) maintains accuracy.

Table III shows that our proposed approach reaches 78% in 18 rounds compared to 200 rounds for FedAvg. This implies an 11-fold speedup over FedAvg with negligible increase in communication cost (1.1× with $\tau = 10$). Interestingly, SOSS-FL continues to improve to 81% (same as the centralized SGD), while FedAvg plateaus at $\sim$ 78%, and original Fed-Sophia cross-over and do not even reaches 78%. Finally, although 3-Matrix Fed-Sophia is equally fast to our approach, it incurs almost twice the communication cost.

E. Impact of Quantization

SOSS-FL applies layer-wise stochastic quantization, as detailed in Section VI), to limit communication cost while preserving reconstruction fidelity. We first investigate the effect of quantization on the different benchmarks with fixed number of quantization bits $B = 8$ in Fig. 8. Figure shows that quantized FedAvg (Blue) collapses to $\approx 70\%$ accuracy, whereas SOSS-FL (Orange) maintains $\approx 81\%$, effectively matching the unquantized baseline. This demonstrates that transmitting the momentum vector is substantially *more robust* to quantization noise than transmitting raw model weights.

Next, Fig. 9 explores the test accuracy performance of SOSS-FL with varying bit-widths ($B \in \{4, 6, 8\}$). Both 6-bit (Blue) and 8-bit (Green) quantization curves match full-precision accuracy ($\approx$ 81%). Nevertheless, at 4-bit precision (Red), performance degrades to $\approx$ 75% with high variance. Thus, $B = 6$ results in the highest reduction in bit-width 5.3× without incurring any loss in test accuracy.

Now, with $B = 6$ and $\tau = 10$, we evaluate the communication cost of SOSS-FL (which achieves 81% accuracy) versus the full-precision FP32 FedAvg (which achieves 78% accuracy). For our proposed scheme, the communication cost is $2Bd(1 + \frac{1}{\tau}) = 13.2d$ per round. FedAvg needs $2Bd = 64d$ per round. Hence, our proposed approach achieves a compression

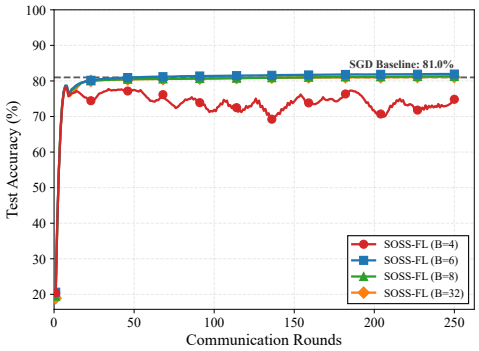


Fig. 9. Sensitivity to quantization levels. 6- and 8-bit match full precision; 4-bit suffers performance degradation.

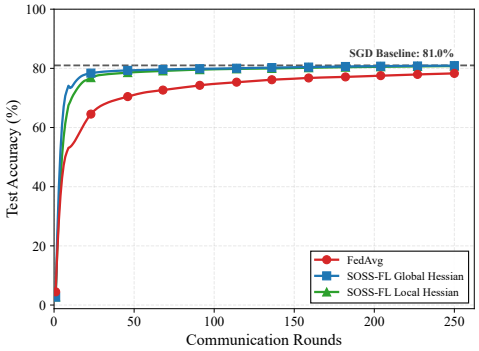


Fig. 10. Impact of Hessian synchronization. *SOSS-FL Local Hessian* (Green) uses $\mathbf{h}_s$ for reconstruction but updates locally with drifting $\mathbf{h}_k$, resulting in performance degradation compared to the fully synchronized method (Blue). This demonstrates that periodic Hessian updates are critical for guiding local training.

ratio of $\approx 4.85\times$. This calculation confirms that layer-wise quantization combined with weight reconstruction provides significant communication savings, achieves better accuracy, while providing impressive training acceleration.

F. Hessian Synchronization Ablation

Fig. 10 evaluates the necessity of synchronizing the Hessian for local training. In this ablation study, the reconstruction step is performed correctly using the server Hessian $\mathbf{h}_s$. However, during the local training phase (Phase 2), clients rely solely on their local Hessian estimates $\mathbf{h}_k$ without periodically synchronizing with the server state. Fig. 10 shows that relying only on local Hessian estimates incurs a loss with respect to periodic Hessian synchronization with the server, signifying the importance of Hessian synchronization in ensuring stable second-order updates.

VIII. Conclusion

In this paper, we proposed SOSS-FL, a communication-efficient framework that stabilizes second-order federated learning by synchronizing optimizer states rather than model weights. This approach effectively resolves the *cross-over divergence* caused by state mismatch in non-IID environments. Empirical results on Fashion-MNIST confirm that SOSS-FL achieves an $11\times$ convergence speedup over FedAvg, matches the accuracy of full-state synchronization, and reduces communication overhead by 48%. Furthermore, with 6-bit quantization, SOSS-FL delivers a $4.85\times$ compression ratio with robust performance, establishing it as a practical solution for bandwidth-constrained edge networks. Future work will extend this framework to larger-scale models and dynamic client participation.

References

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. Artif. Intell. Statist. (AISTATS)*, Fort Lauderdale, FL, USA, Apr. 2017, pp. 1273–1282.

[2] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, pp. 1–19, Feb. 2019.

[3] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, and C. Miao, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 3, pp. 2031–2063, 3rd Quart., 2020.

[4] P. Kairouz *et al.*, "Advances and open problems in federated learning," *Found. Trends Mach. Learn.*, vol. 14, no. 1–2, pp. 1–210, 2021.

[5] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, "SCAFFOLD: Stochastic controlled averaging for federated learning," in *Proc. 37th Int. Conf. Mach. Learn. (ICML)*, Virtual Event, Jul. 2020, pp. 5132–5143.

[6] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," in *Proc. NIPS Private Multi-Party Mach. Learn. Workshop*, Barcelona, Spain, Dec. 2016.

[7] A. S. Berahas, R. Bollapragada, and J. Nocedal, "An adaptive subsampled Newton method for convex optimization," *Math. Program.*, vol. 186, pp. 89–124, Mar. 2021.

[8] R. Crane and F. Roosta, "DINGO: Distributed Newton-type method for gradient-norm optimization," in *Proc. 33rd Adv. Neural Inf. Process. Syst. (NeurIPS)*, Vancouver, BC, Canada, Dec. 2019, pp. 9402–9412.

[9] H. Liu, Z. Li, D. Hall, P. Liang, and T. Ma, "Sophia: A scalable stochastic second-order optimizer for language model pre-training," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, Vienna, Austria, May 2024.

[10] A. Elbakary, C. B. Issaid, M. Shehab, K. Seddik, T. ElBatt, and M. Bennis, "Fed-Sophia: A communication-efficient second-order federated learning algorithm," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Denver, CO, USA, Jun. 2024, pp. 1–6.

[11] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *Proc. 3rd Conf. Mach. Learn. Syst. (MLSys)*, Austin, TX, USA, Mar. 2020, pp. 429–450.

[12] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "FedDANE: A federated Newton-type method," in *Proc. 53rd Asilomar Conf. Signals, Syst., Comput.*, Pacific Grove, CA, USA, Nov. 2019, pp. 1227–1231.

[13] S. Wang, F. Roosta-Khorasani, P. Xu, and M. W. Mahoney, "GIANT: Globally improved approximate Newton method for distributed optimization," in *Proc. 32nd Adv. Neural Inf. Process. Syst. (NeurIPS)*, Montreal, QC, Canada, Dec. 2018, pp. 2332–2342.

[14] A. Ghalkha, C. B. Issaid, and M. Bennis, "Scalable and resource-efficient second-order federated learning via over-the-air aggregation," *arXiv preprint arXiv:2403.03080*, 2024.

[15] M. Chen, Z. Yang, W. Saad, C. Yin, H. V. Poor, and S. Cui, "A joint learning and communications framework for federated learning over wireless networks," *IEEE Trans. Wireless Commun.*, vol. 20, no. 1, pp. 269–283, Jan. 2021.

[16] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 50–60, May 2020.

[17] Q. Yang, Y. Liu, Y. Cheng, Y. Kang, T. Chen, and H. Yu, *Federated Learning*. Morgan & Claypool Publishers, 2019.

[18] A. Elgabli, C. B. Issaid, A. S. Bedi, K. Rajawat, M. Bennis, and V. Aggarwal, "FedNew: A communication-efficient and privacy-preserving Newton-type method for federated learning," in *Proc. 39th Int. Conf. Mach. Learn. (ICML)*, Baltimore, MD, USA, Jul. 2022, pp. 5885–5907.

[19] D. Alistarh, D. Grubic, J. Z. Li, R. Tomioka, and M. Vojnovic, "QSGD: Communication-efficient SGD via gradient quantization and encoding," in *Proc. 31st Adv. Neural Inf. Process. Syst. (NIPS)*, Long Beach, CA, USA, Dec. 2017, pp. 1709–1720.