FedPM: Federated Learning Using Second-order Optimization with Preconditioned Mixing of Local Parameters

Hiro Ishii[1], **Kenta Niwa**[2], **Hiroshi Sawada**[2], **Akinori Fujino**[2], **Noboru Harada**[2], **Rio Yokota**[1]

[1]Institute of Science Tokyo
[2]NTT Communication Science Laboratories
{ ishii.h, rioyokota }@rio.scrc.iir.isct.ac.jp, { kenta.niwa, hrsh.sawada, harada.noboru }@ntt.com

Abstract

We propose Federated Preconditioned Mixing (FedPM), a novel Federated Learning (FL) method that leverages second-order optimization. Prior methods–such as LocalNewton, LTDA, and FedSophia–have incorporated second-order optimization in FL by performing iterative local updates on clients and applying *simple mixing* of local parameters on the server. However, these methods often suffer from drift in local preconditioners, which significantly disrupts the convergence of parameter training, particularly in heterogeneous data settings. To overcome this issue, we refine the update rules by decomposing the ideal second-order update–computed using globally preconditioned global gradients–into parameter mixing on the server and local parameter updates on clients. As a result, our FedPM introduces *preconditioned mixing* of local parameters on the server, effectively mitigating drift in local preconditioners. We provide a theoretical convergence analysis demonstrating a superlinear rate for strongly convex objectives in scenarios involving a single local update. To demonstrate the practical benefits of FedPM, we conducted extensive experiments. The results showed significant improvements with FedPM in the test accuracy compared to conventional methods incorporating simple mixing, fully leveraging the potential of second-order optimization.

Code — https://github.com/rioyokotalab/fedpm
Extended version — http://arxiv.org/abs/2511.09100

1 Introduction

Federated Learning (FL) is a distributed learning paradigm in which multiple clients collaboratively train a global parameter without sharing their local datasets (Kairouz et al. 2021; Konečný et al. 2017, 2016; McMahan et al. 2017), thereby offering benefits in privacy (Bonawitz et al. 2017; Geyer, Klein, and Nabi 2018; Ozdayi, Kantarcioglu, and Gel 2021), scalability (Bonawitz et al. 2019; Li et al. 2020; Oldenhof et al. 2023), and communication efficiency (Lee, Zhang, and Avestimehr 2023; Zang et al. 2024).

As summarized in Table 1, FL methods can be broadly classified into four categories–FOGM, FOPM, SOGM, and SOPM–based on the combination of two key factors: (i) whether the update rule employs First-Order (FO) or Second-Order (SO) optimization methods, and (ii) whether the server-side processing relies on local Gradient Mixing (GM) or local Parameter Mixing (PM). FO methods primarily rely on the gradient–the first derivative of the loss function–to guide parameter updates. In contrast, SO methods also leverage the Hessian (the second derivative), which captures the curvature of the loss landscape. By utilizing this curvature information, SO methods can make more informed updates, which often accelerates the training process. Well-known Parallel SGD (PSGD; also referred to as Minibatch SGD or Distributed SGD) (Collins et al. 2022; Woodworth et al. 2020), which updates the global parameter through simple mixing of local gradients, falls into FOGM. In contrast, FedAvg (McMahan et al. 2017) is classified as FOPM, as the global parameter is obtained by averaging local updated parameters, where each client may perform multiple local updates before communication. Second-order counterparts–SOGM and SOPM–have also been explored. Notably, SOPM methods such as LocalNewton (Gupta et al. 2021), Linear-Time Diagonal Approximation (LTDA) (Sen, Mohan, and Qin 2023), and FedSophia (Elbakary et al. 2024) aim to potentially accelerate and improve the efficiency of training by leveraging both (i) curvature information (i.e., Hessian) and (ii) multiple local updates on each client, thereby reducing communication overhead.

In conventional SOPM methods, server-side processing typically consists of simple mixing of local parameters transmitted from local clients. However, this approach suffers from a fundamental limitation: it does not correspond to a true global second-order optimization. As detailed in Sec. 2.2, our preliminary analysis revealed that such simple mixing results in a global parameter update based on the sum of locally preconditioned local gradients, rather than a globally preconditioned global gradient. This mismatch introduces a critical bottleneck in federated learning, particularly under heterogeneous datasets. In heterogeneous data settings, the local curvature–captured by each client's Hessian–often poorly approximates the global curvature. Consequently, this reliance on mismatched local curvature information can lead to a drift in local preconditioners, which ultimately hinders convergence and prevents the full potential of second-order optimization from being realized, particularly in highly heterogeneous data settings.

To enable global parameter updates that fully lever-

Method category	Representative methods	Update rules	Server-side processing	Global FO opt.	Global SO opt.	Multiple local updates
FOGM	PSGD, Minibatch SGD	Eq. (1)	Global parameter update using simple mixing of local gradients	✓	–	–
FOPM	FedAvg, FedProx, SCAFFOLD	Eq. (2)	Simple mixing of local parameters	✓	–	✓
SOGM	FedNL, FedNew, FedNS	Eq. (4)	Global parameter update using simple mixing of local gradients and preconditioning matrices	–	✓	–
SOPM	LocalNewton, LTDA, FedSophia	Eq. (5)	Simple mixing of local parameters	–	$-^{(*1)}$	✓
	FedPM (ours)	Eq. (10)	**Preconditioned mixing** of local parameters	–	$\checkmark^{(*2)}$	✓

Table 1: Comparison of representative FL methods. All methods can be classified into four categories based on the combination of two key factors: (i) optimization manner: either First-Order (FO) or Second-Order (SO) methods, and (ii) server-side processing: either Gradient Mixing (GM) or Parameter Mixing (PM). Global FO/SO opt. correspond to update rules (3) and (6), respectively. While details are explained in Sec. 2, we reveal that existing SOPM methods cannot achieve global SO optimization due to the simple mixing of local parameters (^{*1}). In contrast, our proposed method, FedPM, employs *preconditioned mixing* of local parameters on the server, which enables global SO optimization with single local update setting (^{*2}).

age second-order information, we propose a novel SOPM method termed Federated Preconditioned Mixing (FedPM). As outlined above, conventional SOPM methods rely on simple mixing of local parameters on the server, yielding global parameter updates using locally preconditioned local gradients. In contrast, our approach derives client-server update rules by decomposing an ideal global second-order optimization using globally preconditioned global gradients. This formulation leads to a core innovation: replacing simple mixing on the server with *preconditioned mixing* of local parameters–a curvature-aware local parameter mixing. By incorporating this preconditioned mixing, FedPM effectively captures global curvature information, leading to improved convergence even in heterogeneous data settings. As highlighted in Table 1, FedPM is the only SOPM method that achieves global second-order optimization while allowing multiple local updates. Our contributions are summarised as follows:

Derivation of curvature-aware SOPM method (FedPM): We derive FedPM, a novel FL algorithm featuring preconditioned mixing on the server, which aligns local updates with the global curvature, expecting robustness towards heterogeneous data settings (Sec. 3.1).

Convergence analysis: A theoretical convergence rate of FedPM is given for a limited condition (strongly convex objective with single local update). It demonstrates a superlinear convergence rate (Sec. 3.2).

Practical implementation: To train large-scale models (e.g., non-convex deep learning models), an efficient preconditioner approximation method, FOOF (Benzing 2022), is employed (Sec. 3.3).

Empirical superiority: We empirically demonstrated that FedPM significantly outperformed existing FO and SO methods in convergence speed and best test accuracy, especially on heterogeneous data settings (Sec. 4).

2 Related Works

2.1 First-Order (FO) Distributed Optimization

We categorize FO distributed optimization methods into two categories: FOGM and FOPM.

FOGM methods, including Parallel SGD (PSGD) (Collins et al. 2022; Woodworth et al. 2020), update the global parameter $\boldsymbol{\theta} \in \mathbb{R}^d$ by aggregating local gradients from each client in every communication round, as

$$\begin{aligned} &\text{Client: } \boldsymbol{g}_i^{(t)} = \nabla f_i(\boldsymbol{\theta}^{(t)}) \qquad \forall i \in \{1, \dots, N\}, \quad (1) \\ &\text{Server: } \boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \frac{\eta}{N} \sum_{i=1}^{N} \boldsymbol{g}_i^{(t)}, \end{aligned}$$

where the initial parameter $\boldsymbol{\theta}^{(0)}$ is given, f_i is a twice-differentiable local loss function, $\nabla f_i(\boldsymbol{\theta}^{(t)})$ represents the local gradient computed using local datasets, N is the number of clients, η is the learning rate, and $t \in \{0, \dots, T-1\}$ denotes the communication round index. This simple method, however, incurs high communication costs.

To improve communication-efficiency, FOPM methods like FedAvg (McMahan et al. 2017), FedProx (Li et al. 2020), and SCAFFOLD (Karimireddy et al. 2020) allow clients to perform multiple local updates ($K \geq 1$) and only transmit the resulting parameters:

$$\begin{aligned} &\text{Client: } \boldsymbol{\theta}_i^{(t,0)} = \boldsymbol{\theta}^{(t)}, \qquad \forall i \in \{1, \dots, N\}, \quad (2) \\ &\boldsymbol{\theta}_i^{(t,k+1)} = \boldsymbol{\theta}_i^{(t,k)} - \eta \nabla f_i(\boldsymbol{\theta}_i^{(t,k)}), \ \forall k \in \{0, \dots, K-1\}, \\ &\text{Server: } \boldsymbol{\theta}^{(t+1)} = \underbrace{\frac{1}{N} \sum_{i=1}^{N} \boldsymbol{\theta}_i^{(t,K)}}_{\text{simple mixing}}. \end{aligned}$$

In the case of a single local update ($K = 1$), the client-server update rules in FOGM and FOPM are combined, resulting in an equivalent global update for both FOGM and

FOPM, as follows:

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \eta \underbrace{\left(\frac{1}{N} \sum_{i=1}^{N} \nabla f_i(\boldsymbol{\theta}^{(t)}) \right)}_{\text{global gradient}}. \tag{3}$$

Equation (3) represents an ideal FO optimization, where the global parameter is updated using global gradient, which is feasible when all datasets are centrally aggregated.

2.2 Second-Order (SO) Distributed Optimization

Similar to FO distributed optimization, SO distributed optimization methods can be categorized into SOGM and SOPM.

SOGM methods including FedNL (Safaryan et al. 2022), FedNew (Elgabli et al. 2022), and FedNS (Li, Liu, and Wang 2024) are communication-intensive, requiring clients to transmit local gradients and Hessians ($\boldsymbol{P}_i^{(t)}$) every round:

$$\begin{aligned} &\text{Client: } \boldsymbol{g}_i^{(t)} = \nabla f_i(\boldsymbol{\theta}^{(t)}), \quad \boldsymbol{P}_i^{(t)} = \nabla^2 f_i(\boldsymbol{\theta}^{(t)}), \\ &\qquad\qquad\qquad\qquad\qquad\qquad \forall i \in \{1, \dots, N\}, \\ &\text{Server: } \boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \eta \left(\frac{1}{N} \sum_{i=1}^{N} \boldsymbol{P}_i^{(t)} \right)^{-1} \left(\frac{1}{N} \sum_{i=1}^{N} \boldsymbol{g}_i^{(t)} \right). \end{aligned} \tag{4}$$

The communication-efficient alternative, SOPM, includes methods such as LocalNewton (Gupta et al. 2021), LTDA (Sen, Mohan, and Qin 2023), and FedSophia (Elbakary et al. 2024), where each client performs local SO updates, while the server simply mixes the local parameters, as

$$\begin{aligned} &\text{Client: } \boldsymbol{\theta}_i^{(t,0)} = \boldsymbol{\theta}^{(t)}, \\ &\qquad \boldsymbol{\theta}_i^{(t,k+1)} = \boldsymbol{\theta}_i^{(t,k)} - \eta \left(\nabla^2 f_i(\boldsymbol{\theta}_i^{(t,k)}) \right)^{-1} \nabla f_i(\boldsymbol{\theta}_i^{(t,k)}), \\ &\qquad\qquad \forall i \in \{1, \dots, N\}, \forall k \in \{0, \cdots, K-1\}, \\ &\text{Server: } \boldsymbol{\theta}^{(t+1)} = \underbrace{\frac{1}{N} \sum_{i=1}^{N} \boldsymbol{\theta}_i^{(t,K)}}_{\text{simple mixing}}. \end{aligned} \tag{5}$$

As shown in the following analysis, SOGM and an instance (5) of SOPM are not equivalent, in contrast to the equivalence observed in their FO counterparts. By combining client-server update rules in (4) under a single-local update setting ($K = 1$), the SOGM updates simplifies to

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \eta \underbrace{\left(\frac{1}{N} \sum_{i=1}^{N} \nabla^2 f_i(\boldsymbol{\theta}^{(t)}) \right)^{-1}}_{\text{global preconditioner}} \underbrace{\left(\frac{1}{N} \sum_{i=1}^{N} \nabla f_i(\boldsymbol{\theta}^{(t)}) \right)}_{\text{global gradient}}. \tag{6}$$

Eq. (6) is an ideal SO optimization, where the global parameter is updated using a globally preconditioned global gradient, which is feasible when all datasets are centrally aggregated. In contrast, the SOPM update in (5) under $K = 1$ can be simplified as follows:

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \frac{\eta}{N} \sum_{i=1}^{N} \underbrace{\left(\nabla^2 f_i(\boldsymbol{\theta}^{(t)}) \right)^{-1}}_{\text{local preconditioner}} \underbrace{\nabla f_i(\boldsymbol{\theta}^{(t)})}_{\text{local gradient}}. \tag{7}$$

In (7), the global parameter is updated using the average of locally preconditioned local gradients. This differs from an ideal SO optimization in (6), which employs a globally preconditioned global gradient.

We showed that conventional SOPM (5) captures local curvatures instead of global curvature. This mismatch becomes problematic in FL, where data heterogeneity causes local preconditioners to be poor estimates of the global one, leading to suboptimal convergence.

3 Proposed Method

To capture global curvature–rather than local curvatures–for faster and more stable global parameter optimization, we propose FedPM, a novel SOPM method. Our goal is to design an SOPM method in which the combination of client-server update rules yields the update in (6), which captures global curvature. To achieve this, our FedPM is derived by decomposing (6) into client-server update rules, as detailed in Sec. 3.1. A theoretical convergence analysis of FedPM under a certain condition is presented in Sec. 3.2. To address computational overheads, an efficient preconditioner approximation technique is introduced in Sec. 3.3, enabling the use of large-scale models (e.g., Deep Neural Networks (DNNs)) as evaluated in Sec. 4.

3.1 Derivation of FedPM

To address the issues of conventional SOPM methods discussed in Sec. 2.2, our core idea is to reformulate the global SO optimization in (6) into local parameter updates on clients and parameter mixing on the server. For this goal, (6) is reformulated as follows:

$$\begin{aligned} \boldsymbol{\theta}^{(t+1)} &= \boldsymbol{\theta}^{(t)} - \eta (\boldsymbol{P}^{(t)})^{-1} \boldsymbol{g}^{(t)} \\ &= (\boldsymbol{P}^{(t)})^{-1} \boldsymbol{P}^{(t)} \boldsymbol{\theta}^{(t)} - \eta (\boldsymbol{P}^{(t)})^{-1} \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{P}_i^{(t)} (\boldsymbol{P}_i^{(t)})^{-1} \boldsymbol{g}_i^{(t)} \\ &= \frac{1}{N} \sum_{i=1}^{N} \left((\boldsymbol{P}^{(t)})^{-1} \boldsymbol{P}_i^{(t)} \boldsymbol{\theta}^{(t)} - \eta (\boldsymbol{P}^{(t)})^{-1} \boldsymbol{P}_i^{(t)} (\boldsymbol{P}_i^{(t)})^{-1} \boldsymbol{g}_i^{(t)} \right) \\ &= \frac{1}{N} \sum_{i=1}^{N} (\boldsymbol{P}^{(t)})^{-1} \boldsymbol{P}_i^{(t)} \left(\boldsymbol{\theta}^{(t)} - \eta (\boldsymbol{P}_i^{(t)})^{-1} \boldsymbol{g}_i^{(t)} \right), \end{aligned} \tag{8}$$

where $\boldsymbol{g}_i^{(t)} = \nabla f_i(\boldsymbol{\theta}^{(t)}), \boldsymbol{g}^{(t)} = \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{g}_i^{(t)}, \boldsymbol{P}_i^{(t)} = \nabla^2 f_i(\boldsymbol{\theta}^{(t)}), \boldsymbol{P}^{(t)} = \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{P}_i^{(t)}$. A straightforward decomposition of this leads to the update rules of our FedPM, as follows:

[FedPM with single local update $(K = 1)$**]**

$$\begin{aligned} &\text{Client: } \boldsymbol{\theta}_i^{(t)} = \boldsymbol{\theta}^{(t)}, \boldsymbol{P}_i^{(t)} = \nabla^2 f_i(\boldsymbol{\theta}^{(t)}), \ \forall i \in \{1, \dots, N\}, \\ &\qquad \boldsymbol{\theta}_i^{(t+1)} = \boldsymbol{\theta}_i^{(t)} - \eta (\boldsymbol{P}_i^{(t)})^{-1} \nabla f_i(\boldsymbol{\theta}^{(t)}), \end{aligned} \tag{9}$$

$$\text{Server: } \boldsymbol{P}^{(t)} = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{P}_i^{(t)},$$

$$\boldsymbol{\theta}^{(t+1)} = \underbrace{\frac{1}{N}\sum_{i=1}^{N} (\boldsymbol{P}^{(t)})^{-1}\boldsymbol{P}_i^{(t)}\boldsymbol{\theta}_i^{(t+1)}}_{\text{preconditioned mixing}},$$

where local parameter mixing on the server is referred to as *preconditioned mixing*. A natural extension of (9), allowing multiple local parameter updates, results in the following:

[FedPM with multiple local updates $(K > 1)$**]**

$$\begin{aligned}
\text{Client: } \boldsymbol{\theta}_i^{(t,0)} &= \boldsymbol{\theta}^{(t)}, \qquad \forall i \in \{1,\dots,N\}, \quad (10)\\
\boldsymbol{P}_i^{(t,k)} &= \nabla^2 f_i(\boldsymbol{\theta}_i^{(t,k)}), \qquad \forall k \in \{0,\cdots,K-1\},\\
\boldsymbol{\theta}_i^{(t,k+1)} &= \boldsymbol{\theta}_i^{(t,k)} - \eta(\boldsymbol{P}_i^{(t,k)})^{-1}\nabla f_i(\boldsymbol{\theta}^{(t,k)}),
\end{aligned}$$

$$\text{Server: } \boldsymbol{P}^{(t)} = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{P}_i^{(t,K-1)},$$

$$\boldsymbol{\theta}^{(t+1)} = \underbrace{\frac{1}{N}\sum_{i=1}^{N} (\boldsymbol{P}^{(t)})^{-1}\boldsymbol{P}_i^{(t,K-1)}\boldsymbol{\theta}_i^{(t,K)}}_{\text{preconditioned mixing}}.$$

The key feature of FedPM lies in its use of preconditioned mixing, which enables effective capture of global curvature information. However, unlike conventional SOPM methods (e.g., LocalNewton, LTDA, and FedSophia), FedPM incurs additional communication costs as it requires transmitting not only local parameters but also local preconditioners. Nevertheless, even with this additional communication overhead, FedPM is preferable in heterogeneous data settings. By capturing global curvature–rather than relying on statistically biased local curvatures–FedPM can achieve faster and more stable convergence even under heterogeneous data distributions.

3.2 Theoretical Convergence Analysis

We provide a convergence analysis of FedPM under a limited condition. In line with most convergence proofs for second-order methods (Safaryan et al. 2022; Elgabli et al. 2022; Li, Liu, and Wang 2024), our analysis assumes the loss function is strongly convex. Furthermore, the analysis is confined to the case of a single local update ($K = 1$), as the drift in local preconditioners during multiple updates complicates a rigorous convergence analysis. The analysis also disregards stochastic noise by assuming full local gradients and Hessians are available.

Theorem 1 ((Informal) Convergence rate of FedPM under single local update)**.** *Under the assumptions of strong convexity and Hessian smoothness, and given an initial parameter condition that implicitly assumes sufficiently close to the optimal solution, the FedPM algorithm with a single local update ($K = 1$) as defined in (9) achieves a superlinear convergence rate for the global parameter.*

Discussion of Theorem 1: This theoretical result demonstrates the potential of FedPM to achieve a superlinear convergence rate, providing a theoretical validation of FedPM's faster and more stable convergence. Our analysis builds upon the strategy used for FedNL, since FedPM with a single local update ($K = 1$) follows the same global parameter update rule given in (6). A subtle but important distinction lies in the use of the most recent Hessian for preconditioning, which is a more direct application of Newton's method than the stale Hessian used in the original FedNL's proof in (Safaryan et al. 2022). Based on the constraints outlined above, our numerical experiments in Sec. 4 are designed with two objectives: Test 1 in Sec. 4.1 aims to empirically validate Theorem 1, using a strongly convex loss function with full gradient under single local update. In contrast, Test 2 in Sec. 4.2 focuses on more practical scenarios, evaluating FedPM on non-convex DNN model training.

Method	Construction	Inverse	Communication
FedPM	$\mathcal{O}(Md^2)$	$\mathcal{O}(d^3)$	$\mathcal{O}(d^2)$
FedPM w/ FOOF	$\mathcal{O}(Md)$	$\mathcal{O}(d\sqrt{d/L})$	$\mathcal{O}(d)$

Table 2: Comparison of computation and communication costs between FedPM (with full Hessian) and FedPM with FOOF approximation.

3.3 Preconditioner Approximation Method

To apply FedPM to DNN models, where direct computation of the full Hessian matrix is impractical, we employ an efficient preconditioner approximation. A popular strategy is to leverage the Fisher Information Matrix (FIM) to approximate the Hessian (Amari 1998; Martens 2020). Under common loss functions like cross-entropy, the FIM is equivalent to the Generalized Gauss-Newton (GGN) matrix, a widely used Hessian approximation in deep learning (Martens 2020; Schraudolph 2002).

In this paper, we employ FOOF (Benzing 2022) as an FIM approximation method. FOOF is an effective FIM approximation method, particularly beneficial for DNNs holding a large number of parameters. It simplifies the FIM computation by approximating it with block-diagonal matrices, where each block corresponds to one of L layers of the DNN. To further reduce computational complexity, each block is approximated as an uncentered covariance matrix of inputs of the layer. The update rule for FOOF in a specific layer l of a DNN at the i-th client is given by:

$$\boldsymbol{\theta}_{i,l}^{(t,k+1)} = \boldsymbol{\theta}_{i,l}^{(t,k)} - \eta \,\mathrm{vec}\left(\left(\boldsymbol{A}_{i,l}^{(t,k)}\right)^{-1} \mathrm{vec}^{-1}\left(\boldsymbol{g}_{i,l}^{(t,k)}\right)\right), \tag{11}$$

where $\boldsymbol{\theta}_{i,l}^{(t,k)}$ and $\boldsymbol{g}_{i,l}^{(t,k)}$ represent the parameter and the gradient of the l-th layer of i-th client at k-th local iteration during the t-th communication round, respectively, The matrix $\boldsymbol{A}_{i,l}^{(t,k)}$ represents the uncentered covariance matrix of the inputs (FOOF matrix) of the l-th layer. The operators vec and vec^{-1} denote the operations to vectorize a matrix and revert a vector back to a matrix form, respectively. FOOF is particularly useful in scenarios where balancing computational cost and performance is crucial. The preconditioned mixing

Algorithm 1: Federated Preconditioned Mixing (FedPM)

Server Initialization: Global parameter $\boldsymbol{\theta}^{(0)}$
for $t = 0$ **to** $T-1$ **do**
  Server sends global parameter $\boldsymbol{\theta}^{(t)}$ to all clients.
  for $i = 1$ **to** N **in parallel do**
    $\boldsymbol{\theta}_i^{(t,0)} \leftarrow \boldsymbol{\theta}^{(t)}$
    for $k = 0$ **to** $K-1$ **do**
      if using FOOF approximation **then**
        for $l = 1$ **to** L **do**
          Update $\boldsymbol{\theta}_{i,l}^{(t,k+1)}$ per Eq. (11).
        end for
      else
        $\boldsymbol{P}_i^{(t,k)} \leftarrow \nabla^2 f_i(\boldsymbol{\theta}_i^{(t,k)})$.
        $\boldsymbol{\theta}_i^{(t,k+1)} \leftarrow \boldsymbol{\theta}_i^{(t,k)} - \eta(\boldsymbol{P}_i^{(t,k)})^{-1}\nabla f_i(\boldsymbol{\theta}_i^{(t,k)})$.
      end if
    end for
    Client i sends $\boldsymbol{\theta}_i^{(t,K)}$ and preconditioner(s) (i.e., $\boldsymbol{P}_i^{(t,K-1)}$ or $\{\boldsymbol{A}_{i,l}^{(t,K-1)}\}_{l=1}^{L}$) to the server.
  end for
  Server aggregates:
  if using FOOF approximation **then**
    for $l = 1$ **to** L **do**
      Update $\boldsymbol{\theta}_l^{(t+1)}$ per Eq. (12).
    end for
  else
    $\boldsymbol{P}^{(t)} \leftarrow \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{P}_i^{(t,K-1)}$.
    $\boldsymbol{\theta}^{(t+1)} \leftarrow \frac{1}{N}\sum_{i=1}^{N} (\boldsymbol{P}^{(t)})^{-1}\boldsymbol{P}_i^{(t,K-1)}\boldsymbol{\theta}_i^{(t,K)}$.
  end if
end for

at the center server for the FOOF approximation is given by:

$$\boldsymbol{\theta}_l^{(t+1,0)} = \mathrm{vec}\left(\left(\frac{1}{N}\sum_{i=1}^{N}\boldsymbol{A}_{i,l}^{(t,K-1)}\right)^{-1}\left(\frac{1}{N}\sum_{i=1}^{N}\boldsymbol{A}_{i,l}^{(t,K-1)}\,\mathrm{vec}^{-1}\left(\boldsymbol{\theta}_{i,l}^{(t,K)}\right)\right)\right). \tag{12}$$

The computational and communication costs of the FedPM with the FOOF approximation, compared to FedPM without any preconditioner approximation, are summarized in Table 2. In this comparison, the model architecture is limited to the L-layer fully connected neural network with each layer having consistent input and output dimensions of $\sqrt{d/L}$, and M denotes the number of data samples used to compute the matrix. The table distinguishes the time complexities for constructing and inverting matrices, and the space complexity for storing and transmitting FOOF matrices. Compared to the FedPM using full preconditioning matrices without approximation, the computational and communication costs are drastically reduced. The general procedure for FedPM, which accommodates both the full Hessian and its FOOF approximation, is detailed in Algorithm 1. The effectiveness of this approximation in FedPM is empirically evaluated in the experiments presented in Sec. 4.2.

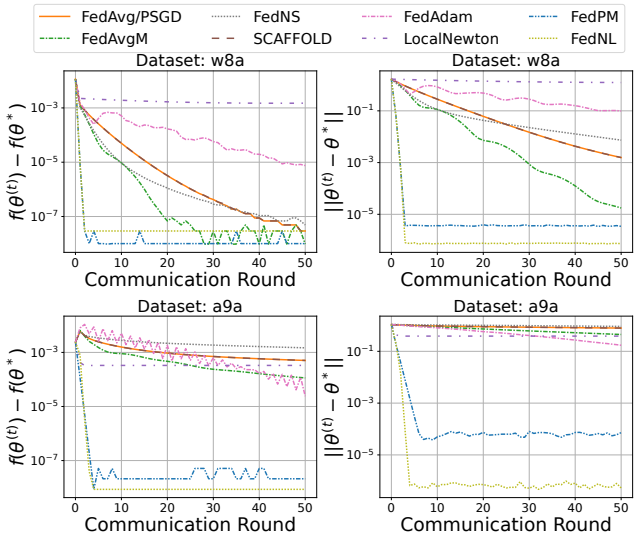

Figure 1: Convergence curves for Test 1 ($K = 1$) using global parameter on w8a (top) and a9a (bottom). The left column displays the difference in function output at each round and at the optimal solution. The right column depicts the L2 norm of the difference between the parameter at each round and the optimal solution.

4 Numerical Experiments

To empirically examine the performance of the proposed FedPM, we prepared two tests: (Test 1) a strongly convex model in Sec. 4.1 and (Test 2) non-convex DNN models in Sec. 4.2.

4.1 Test 1 Using a Strongly Convex Model

Experimental setups

Datasets, model, and network configuration: As a toy experiment designed to satisfy the assumptions used in the theoretical analysis in Sec. 3.2, we employed a strongly convex model: logistic regression with squared L_2 regularization. The loss function is defined as $f(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N} f_i(\boldsymbol{\theta})$, where $f_i(\boldsymbol{\theta}) = \frac{1}{M}\sum_{j=1}^{M}\log\left(1+\exp(-y_{ij}\boldsymbol{x}_{ij}^\top\boldsymbol{\theta})\right)+\frac{\lambda}{2}\|\boldsymbol{\theta}\|^2$ with $\{\boldsymbol{x}_{ij}, y_{ij}\}_{j\in[M]}$ representing data samples held by the i-th client. We used the w8a and a9a datasets from LibSVM (Chang and Lin 2011). The network configuration involves a central server with $N = 142$ clients for w8a and 80 clients for a9a communicating with single local update ($K = 1$) for $T = 50$ communication rounds. The dataset was evenly and homogeneously distributed, with each client assigned 350 data samples for w8a ($d = 300$) and 407 data samples for a9a ($d = 123$).

Comparison methods: We prepared seven comparison methods for evaluation, along with our proposed method: FedAvg/PSGD (Collins et al. 2022; McMahan et al. 2017), FedAvgM (Hsu, Qi, and Brown 2019), SCAFFOLD (Karimireddy et al. 2020), FedAdam (Reddi et al. 2021), FedNS (Li, Liu, and Wang 2024), FedNL (Safaryan et al. 2022), LocalNewton (Gupta et al. 2021), and proposed FedPM. We excluded SOPM methods that use diagonal approximations

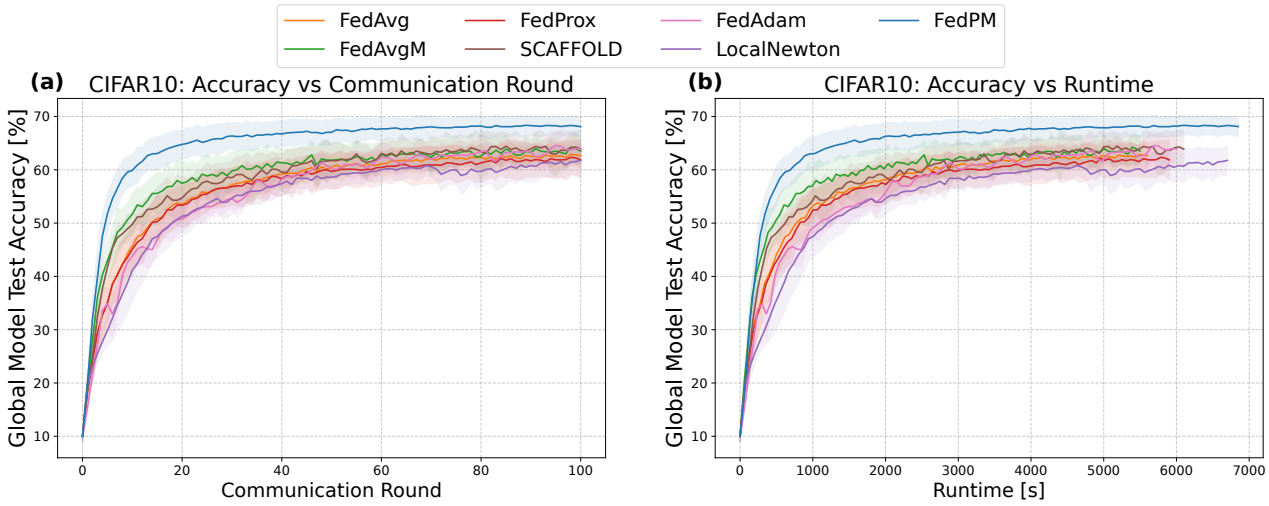

Figure 2: Convergence curves for Test 2 using test accuracy for the global parameter on CIFAR10 classification with heterogeneity level of $\alpha = 0.1$ and 5 local epochs. (a) depicts test accuracy against communication rounds, whereas (b) shows test accuracy against runtime, including communication overhead. The shaded area depicts one standard deviation of results across three different random seeds.

for the Hessian, such as LTDA and FedSophia, as they are designed for scenarios where full Hessian computation is intractable. Since computing the full Hessian is feasible in this experiment, using such approximations would be suboptimal. In Test 1, stochastic gradient computation and FOOF-based preconditioner approximation were not applied to follow the convergence analysis in Sec. 3.2.

Hyperparameter settings: Hyperparameter tuning was conducted to find the best learning rate for each method.

Experimental results Figure 1 illustrates the performance of each method on the w8a and a9a datasets. The left column displays the difference between the function output at each round and the optimal value, $|f(\boldsymbol{\theta}^{(t)}) - f(\boldsymbol{\theta}^*)|$, while the right column depicts the L2 norm of the difference between the parameter at each round and the optimal solution, $\|\boldsymbol{\theta}^{(t)} - \boldsymbol{\theta}^*\|$. The optimal solution $\boldsymbol{\theta}^*$ was determined as the parameter obtained after 20 iterations of standard Newton's method using all data samples. The initial parameter was drawn from a normal distribution centered on $\boldsymbol{\theta}^*$ with a standard deviation of 0.1. This choice ensures that the initial parameter is sufficiently close to the optimal solution, which is a necessary condition for the validity of the convergence analysis. In these experiments, both FedPM and FedNL significantly outperformed other methods, consistently achieving smaller function output gaps and parameter distances, thus indicating more efficient convergence. Notably, the rapid, accelerating decrease in the parameter distance for FedPM and FedNL shown in the right-hand plots is consistent with the superlinear convergence rate shown in Theorem 1. Although FedNL and FedPM with $K = 1$ are equivalent in global parameter updates, minor discrepancies were observed in the outcomes. These differences may be attributed to precision errors resulting from variations in their respective update procedures.

4.2 Test 2 Using Non-Convex DNN Models

Experimental setups

Datasets, models, and network configuration. To empirically assess the effectiveness of the proposed FedPM with non-convex DNN models, we conducted tests using two image classification benchmarks: (T1) CIFAR10[1] with simple CNN as in (Li, He, and Song 2021; Luo et al. 2021), and (T2) CIFAR100 with ResNet18 (He et al. 2016), where BatchNorm layers (Ioffe and Szegedy 2015) were replaced by GroupNorm layers (Wu and He 2018) to enhance robustness against data heterogeneity. The center server and $N = 10$ clients can communicate every $\{1, 5, 10\}$ local update epochs across $T = 100$ communication rounds. Client sampling was not employed in the main paper's experimental results. Data heterogeneity among clients was intentionally induced by varying the Dirichlet distribution concentration parameter α ($\alpha \in \{0.1, 1.0\}$) following the implementation in (Vogels et al. 2021), with a lower α indicating stronger data heterogeneity. Each client holds a different number of data samples. We investigated the impact of the number of local update epochs and the levels of data heterogeneity α on the test accuracy of global parameters.

Comparison methods. In addition to the methods evaluated in Test 1, we added a comparison with FedProx (Li et al. 2020) in Test 2. However, SOGM methods like FedNL and FedNew were found to be computationally prohibitive for these large-scale DNN experiments, failing to complete a sufficient number of rounds within a practical time budget, and were thus excluded from this comparison. Additionally, whereas the local full Hessian was computed as for preconditioner in Test 1, it was approximated using FOOF for LocalNewton and FedPM in Test 2, as detailed in Sec. 3.3.

Hyperparameter settings. The learning rate, the norm

[1]https://www.cs.toronto.edu/~kriz/cifar.html

Method	CIFAR10		CIFAR100	
	$\alpha = 1.0$	$\alpha = 0.1$	$\alpha = 1.0$	$\alpha = 0.1$
FedAvg	70.1±1.3	63.1±2.1	59.8±0.7	52.4±0.9
FedAvgM	72.6±1.5	64.8±3.0	61.2±0.2	54.9±0.6
FedProx	70.1±1.1	62.8±1.3	56.6±0.7	50.7±0.9
SCAFFOLD	71.4±0.5	65.3±2.7	64.7±1.1	58.2±0.6
FedAdam	70.4±1.6	64.6±2.8	58.3±0.3	51.9±0.4
LocalNewton	73.4±1.4	62.4±2.3	**70.2**±0.3	61.9±0.6
FedPM	**74.8**±0.5	**68.6**±2.0	68.4±2.2	**63.1**±0.8

Table 3: Average best test accuracy of global parameter (over three random seeds) for each method on various datasets with $\alpha = 1.0$ or $\alpha = 0.1$ (the smaller, the stronger heterogeneity), where local updates are performed for 5 epochs before each communication. The standard deviation across different seeds is also shown for each. The FOOF approximation was applied to both FedPM and LocalNewton.

value for gradient clipping, the weight decay coefficient (regularization coefficient μ in FedProx), the damping term for second-order optimization methods, and unique hyperparameters for each method such as the momentum value in FedAvgM, were tuned to achieve the highest average of the highest test accuracy accross three different random seeds for each method. Additionally, we computed FOOF matrices only at the end of each round, just before the communication, to further improve efficiency for second-order methods. We empirically observed that this approach did not degrade performance compared to periodically updating the matrices during local training. For FOOF matrix computation, we utilized the full local dataset.

Experimental results Table 3 displays the highest test accuracy of a global parameter achieved by each method across four test settings, which combine two datasets/models with two levels of data heterogeneity. These results demonstrate that our proposed FedPM outperformed all other comparison methods in test accuracy in most cases. Notably, FedPM exhibited superior performance at the stronger heterogeneity condition ($\alpha = 0.1$), while LocalNewton's performance deteriorated significantly as data heterogeneity increased. These empirical findings suggest that preconditioned mixing improves alignment between local and global parameter updates, offering substantial benefits in heterogeneous data settings.

Figure 2 illustrates convergence curves when local parameters are updated for five epochs before each communication under a stronger heterogeneous data setting ($\alpha = 0.1$) on CIFAR10 classification. The results show that FedPM converged faster than other methods in terms of both communication rounds and runtime, thanks to its efficient preconditioner approximation method. This is crucial in FL because FedPM enables the model to reach the desired performance with fewer communication rounds. These findings confirm that FedPM's ability to incorporate global curvature information through preconditioned mixing is useful for improving not only generalization and parameter accuracy but also the credibility of second-order FL in real-world scenarios.

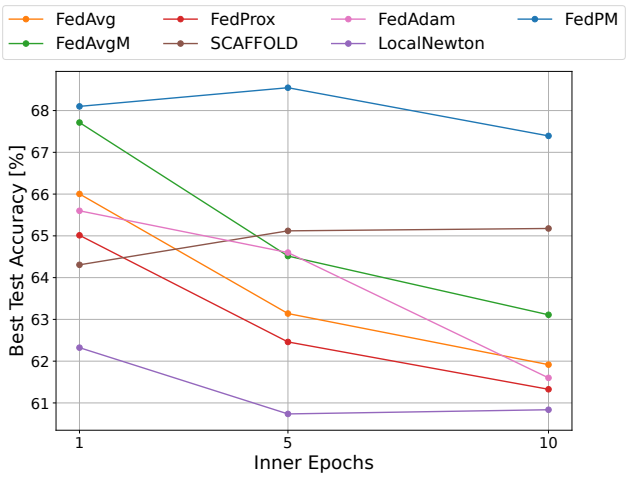

Figure 3: Relationship between the number of multiple local parameter updates and test accuracy for CIFAR10 classification with $\alpha = 0.1$. The plot shows the average highest test accuracy across three seeds. Experiments are conducted with a fixed total of 500 communication rounds achieved by using 1 inner epoch (500 rounds), 5 inner epochs (100 rounds), and 10 inner epochs (50 rounds).

Figure 3 presents the CIFAR10 classification results with $\alpha = 0.1$ on varying number of inner epochs, where the total number of local epochs is fixed to 500 by adjusting the number of communication rounds. Specifically, the three settings correspond to: 500 rounds with 1 local epoch per round, 100 rounds with 5 local epochs per round, and 50 rounds with 10 local epochs per round. For each setting, we report the average of the highest test accuracies achieved by the global model over three different random seeds. The results indicate that FedPM consistently outperforms other methods, even when varying the number of local parameter updates.

5 Conclusion

We introduced FedPM, a novel Federated Learning (FL) method that unlocks the potential of second-order optimization. Unlike conventional Second-Order Parameter Mixing (SOPM) methods that are sensitive to data heterogeneity, FedPM incorporates preconditioned mixing at the server. This aligns parameter updates with the ideal global curvature, enhancing stability and convergence. Our theoretical analysis establishes a superlinear convergence rate for FedPM under strongly convex objectives and a single local update. Extensive experiments confirm its practical superiority: FedPM significantly outperformed state-of-the-art methods on both strongly convex and non-convex deep learning models, demonstrating particular robustness in highly heterogeneous settings. These results validate FedPM as a significant and practical advancement for second-order optimization in real-world federated scenarios.

Acknowledgements

This work was supported by JST SPRING, Japan Grant Number JPMJSP2180. This study was carried out using the TSUBAME4.0 supercomputer at Institute of Science Tokyo.

References

Amari, S.-I. 1998. Natural gradient works efficiently in learning. *Neural computation*, 10(2): 251–276.

Benzing, F. 2022. Gradient descent on neurons and its link to approximate second-order optimization. In *International Conference on Machine Learning*, 1817–1853. PMLR.

Bonawitz, K.; Eichner, H.; Grieskamp, W.; Huba, D.; Ingerman, A.; Ivanov, V.; Kiddon, C.; Konečnỳ, J.; Mazzocchi, S.; McMahan, B.; et al. 2019. Towards federated learning at scale: System design. *Proceedings of machine learning and systems*, 1: 374–388.

Bonawitz, K.; Ivanov, V.; Kreuter, B.; Marcedone, A.; McMahan, H. B.; Patel, S.; Ramage, D.; Segal, A.; and Seth, K. 2017. Practical secure aggregation for privacy-preserving machine learning. In *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 1175–1191.

Chang, C.-C.; and Lin, C.-J. 2011. LIBSVM: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3): 1–27.

Collins, L.; Hassani, H.; Mokhtari, A.; and Shakkottai, S. 2022. Fedavg with fine tuning: Local updates lead to representation learning. *Advances in Neural Information Processing Systems*, 35: 10572–10586.

Elbakary, A.; Issaid, C. B.; Shehab, M.; Seddik, K.; ElBatt, T.; and Bennis, M. 2024. Fed-sophia: A communication-efficient second-order federated learning algorithm. In *ICC 2024-IEEE International Conference on Communications*, 950–955. IEEE.

Elgabli, A.; Issaid, C. B.; Bedi, A. S.; Rajawat, K.; Bennis, M.; and Aggarwal, V. 2022. FedNew: A communication-efficient and privacy-preserving Newton-type method for federated learning. In *International conference on machine learning*, 5861–5877. PMLR.

Geyer, R. C.; Klein, T.; and Nabi, M. 2018. Differentially Private Federated Learning: A Client Level Perspective. arXiv:1712.07557.

Gupta, V.; Ghosh, A.; Derezinski, M.; Khanna, R.; Ramchandran, K.; and Mahoney, M. 2021. LocalNewton: Reducing Communication Bottleneck for Distributed Learning. arXiv:2105.07320.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778.

Hsu, T.-M. H.; Qi, H.; and Brown, M. 2019. Measuring the Effects of Non-Identical Data Distribution for Federated Visual Classification. arXiv:1909.06335.

Ioffe, S.; and Szegedy, C. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, 448–456. PMLR.

Kairouz, P.; McMahan, H. B.; Avent, B.; Bellet, A.; Bennis, M.; Bhagoji, A. N.; Bonawitz, K.; Charles, Z.; Cormode, G.; Cummings, R.; et al. 2021. Advances and open problems in federated learning. *Foundations and trends® in machine learning*, 14(1–2): 1–210.

Karimireddy, S. P.; Kale, S.; Mohri, M.; Reddi, S.; Stich, S.; and Suresh, A. T. 2020. Scaffold: Stochastic controlled averaging for federated learning. In *International conference on machine learning*, 5132–5143. PMLR.

Konečný, J.; McMahan, H. B.; Ramage, D.; and Richtárik, P. 2016. Federated Optimization: Distributed Machine Learning for On-Device Intelligence. arXiv:1610.02527.

Konečný, J.; McMahan, H. B.; Yu, F. X.; Richtárik, P.; Suresh, A. T.; and Bacon, D. 2017. Federated Learning: Strategies for Improving Communication Efficiency. arXiv:1610.05492.

Lee, S.; Zhang, T.; and Avestimehr, A. S. 2023. Layer-wise adaptive model aggregation for scalable federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, 8491–8499.

Li, J.; Liu, Y.; and Wang, W. 2024. Fedns: A fast sketching newton-type algorithm for federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 13509–13517.

Li, Q.; He, B.; and Song, D. 2021. Model-contrastive federated learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 10713–10722.

Li, T.; Sahu, A. K.; Zaheer, M.; Sanjabi, M.; Talwalkar, A.; and Smith, V. 2020. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2: 429–450.

Luo, M.; Chen, F.; Hu, D.; Zhang, Y.; Liang, J.; and Feng, J. 2021. No fear of heterogeneity: Classifier calibration for federated learning with non-iid data. *Advances in Neural Information Processing Systems*, 34: 5972–5984.

Martens, J. 2020. New insights and perspectives on the natural gradient method. *Journal of Machine Learning Research*, 21(146): 1–76.

McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; and y Arcas, B. A. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, 1273–1282. PMLR.

Oldenhof, M.; Ács, G.; Pejó, B.; Schuffenhauer, A.; Holway, N.; Sturm, N.; Dieckmann, A.; Fortmeier, O.; Boniface, E.; Mayer, C.; et al. 2023. Industry-scale orchestrated federated learning for drug discovery. In *Proceedings of the aaai conference on artificial intelligence*, volume 37, 15576–15584.

Ozdayi, M. S.; Kantarcioglu, M.; and Gel, Y. R. 2021. Defending against backdoors in federated learning with robust learning rate. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, 9268–9276.

Reddi, S. J.; Charles, Z.; Zaheer, M.; Garrett, Z.; Rush, K.; Konečný, J.; Kumar, S.; and McMahan, H. B. 2021. Adaptive Federated Optimization. In *International Conference on Learning Representations*.

Safaryan, M.; Islamov, R.; Qian, X.; and Richtarik, P. 2022. FedNL: Making Newton-Type Methods Applicable to Federated Learning. In *International Conference on Machine Learning*, 18959–19010. PMLR.

Schraudolph, N. N. 2002. Fast curvature matrix-vector products for second-order gradient descent. *Neural computation*, 14(7): 1723–1738.

Sen, M.; Mohan, C. K.; and Qin, A. K. 2023. Federated optimization with linear-time approximated hessian diagonal. In *International Conference on Pattern Recognition and Machine Intelligence*, 106–113. Springer.

Vogels, T.; He, L.; Koloskova, A.; Karimireddy, S. P.; Lin, T.; Stich, S. U.; and Jaggi, M. 2021. Relaysum for decentralized deep learning on heterogeneous data. *Advances in Neural Information Processing Systems*, 34: 28004–28015.

Woodworth, B.; Patel, K. K.; Stich, S.; Dai, Z.; Bullins, B.; Mcmahan, B.; Shamir, O.; and Srebro, N. 2020. Is local SGD better than minibatch SGD? In *International Conference on Machine Learning*, 10334–10343. PMLR.

Wu, Y.; and He, K. 2018. Group normalization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 3–19.

Zang, Y.; Xue, Z.; Ou, S.; Chu, L.; Du, J.; and Long, Y. 2024. Efficient asynchronous federated learning with prospective momentum aggregation and fine-grained correction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 16642–16650.