FULLY ASYNCHRONOUS FEDERATED LEARNING WITH FASTER CONVERGENCE FOR LLM REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Federated Learning (FL) has emerged as a transformative paradigm for distributed machine learning, enabling collaborative model training across decentralized devices while preserving data privacy. Concurrently, the advent of Large Language Models (LLMs)—such as GPT, Claude, and Qwen—has redefined natural language understanding and generation. Despite their potential, integrating LLMs into FL frameworks remains challenging; conventional synchronous FL mechanisms frequently suffer from significant communication overhead and idle "straggler" delays, exacerbated by the vast parameter space of LLMs and the inherent hardware heterogeneity of edge devices. To mitigate these inefficiencies, we propose a novel fully asynchronous FL framework specifically optimized for LLM fine-tuning. Our core contribution is a systematic exploration of matrix decomposition and approximation techniques to identify the most effective linear algebraic methods for distributed optimization in asynchronous settings. We evaluate three distinct approaches—Principal Component Analysis (PCA), QR Decomposition with Column Pivoting (QRCP), and CUR Decomposition—through extensive experiments on GPT-2 fine-tuning using the WikiText dataset. Empirical results demonstrate that PCA-based approximation achieves the fastest convergence and competitive accuracy, significantly reducing wall-clock training time while maintaining a performance profile comparable to synchronous baselines.

1 INTRODUCTION

Integrating Federated Learning (FL) with Large Language Models (LLMs) enables users to fine-tune personalized models while strictly preserving data privacy against untrusted third parties or public clouds. However, this integration presents significant technical challenges. Conventional FL methods predominantly rely on synchronous or partially synchronous aggregation, where a central server mandates a quorum of clients to complete local updates before proceeding. This synchronization requirement imposes a substantial bottleneck, leading to systemic inefficiencies and "straggler" delays, which are further exacerbated by the hardware heterogeneity and volatile communication capabilities of edge devices.

The primary objective of this research is to identify an optimal matrix decomposition strategy to resolve the optimization bottlenecks inherent in asynchronous FL frameworks specifically designed for LLMs. To this end, we first propose a fully asynchronous training paradigm that decouples local updates from global synchronization. Under this framework, participating devices perform local training independently and transmit updated parameters to the central server immediately upon completion, thereby maximizing hardware utilization.

Furthermore, we develop an update mechanism designed to handle client heterogeneity while ensuring the stability and convergence of the global model. Our approach accommodates devices with diverse computational and communication profiles, effectively mitigating accuracy degradation. The core focus is centered on identifying optimal matrix approximation solutions to reformulate the optimization problem into a more tractable form. By conducting a rigorous theoretical analysis of the convergence functions, we derive mathematical transformations that simplify complex matrices into manageable scalar representations. This enables an efficient exploration of the trade-offs between model accuracy and system latency. We extend our study across a comprehensive range of approximation techniques to determine the most effective solution for distributed LLM optimization.

In addition, we investigate and develop techniques to mitigate local convergence disparities arising from the non-IID (Independent and Identically Distributed) and imbalanced nature of user data. Since each device operates on a unique data distribution, local models are often prone to representational bias; our method explicitly addresses these local drifts to maintain global model integrity. Finally, we conduct extensive experiments to evaluate the performance of the proposed framework. These empirical evaluations serve to identify the most suitable matrix approximation for federated LLM optimization across diverse real-world scenarios.

2 Proposed Method

2.1 Targeted Data Selection and Personalized Training

To optimize local model performance across heterogeneous clients, we propose a targeted data selection mechanism that explicitly accounts for individual user characteristics and task-specific requirements. This personalized training strategy involves the joint optimization of several key hyperparameters, including the number of local training iterations, fine-tuning configurations, and client selection criteria. By dynamically tailoring these parameters to the unique constraints of each user, we aim to maximize both the efficacy and efficiency of the FL process.

The selection mechanism is formulated as a constrained optimization problem as follows:

$$\begin{array}{ll} \text{minimize} & \mathbb{E}[\, T_{\text{tot}}(\mathbf{q})\,] \\ \text{s.t.} & \mathbb{E}[\, F(\mathbf{w}^R;\mathbf{q})\,] - F^* \le \epsilon \\ & \sum_1^n q_i = 1 \\ & q_i > 0, \forall i \in [n] \end{array}$$

where:

- $\mathbf{q} \in \mathbb{R}^n$ denotes the vector of selection probabilities for each communication round, where client sampling is performed in an unbiased manner based on the distribution $\mathbf{q}$.
- $T_{\text{tot}}(\mathbf{q})$ represents the total wall-clock learning time, modeled as a function of the selection probabilities $\mathbf{q}$. The objective is to minimize the expected training latency.
- $F(\mathbf{w}^R;\mathbf{q})$ denotes the global objective value after R communication rounds, where $\mathbf{w}^R$ are the model parameters at round R. The convergence of the model is implicitly governed by the sampling distribution $\mathbf{q}$.
- F^* represents the optimal value of the objective function, signifying the theoretical lower bound of the loss.
- ϵ is a tolerance parameter that specifies the maximum allowable optimality gap between the expected performance of the trained model and the global optimum.

The first constraint serves as a convergence guarantee, ensuring that the expected performance of the model after R rounds remains within an ϵ-neighborhood of the global optimum F^*. This ensures that the selected subset of clients provides sufficiently informative updates to drive the learning process effectively. The second and third constraints ensure that $\mathbf{q}$ resides on a valid probability simplex, requiring that selection probabilities are normalized and that every client maintains a non-zero probability of participation to prevent premature exclusion of potentially valuable data.

2.2 Solving the Optimization Problem

Following the establishment of the targeted data selection and hyperparameter search components, we formalize the optimization objective and introduce efficient numerical techniques to achieve convergence in the asynchronous setting.

The optimization objective in federated learning is formulated as the minimization of a global objective function, defined as the empirical risk across N distributed clients:

$$\min *\theta F(\theta) = \frac{1}{N} \sum *i = 1^N f_i(\theta) \tag{1}$$

where $F(\theta)$ denotes the global loss, N is the total number of participating clients, and $f_i(\theta)$ represents the local loss function for client i.

Stochastic Gradient Descent (SGD) remains the foundational optimization paradigm for iteratively updating model parameters. The update rule is defined as:

$$\theta^{(t+1)} = \theta^{(t)} - \eta \nabla F\left(\theta^{(t)}\right) \tag{2}$$

where the gradient ∇F is computed via high-dimensional linear algebraic operations. In the context of LLMs, the massive parameter space necessitates efficient gradient computation and communication, often implemented through mini-batch variants to balance throughput and convergence stability.

To analyze the convergence properties of the proposed asynchronous framework, spectral analysis and eigenvalue decomposition are essential. Upon formulating the convergence equations, the computational complexity associated with directly solving inverse matrix problems or Hessian-related operations becomes prohibitive. Furthermore, the non-convex nature of the LLM landscape complicates direct optimization. To address these challenges, we utilize matrix approximation and decomposition techniques. For instance, we leverage Principal Component Analysis (PCA) to project high-dimensional gradient information into lower-dimensional subspaces, thereby mitigating the "curse of dimensionality."

These dimensionality reduction techniques are crucial for managing communication overhead. By compressing the updates before transmission between the edge devices and the central server, we significantly reduce the bandwidth requirements without sacrificing the essential directional information of the gradients.

Beyond standard approximation methods, we investigate two advanced factorization techniques for stable and efficient optimization. First, we employ **QR Decomposition with Column Pivoting (QRCP)**, which provides a numerically stable, well-conditioned factorization even in the presence of rank deficiency. Second, we evaluate the **CUR Decomposition** (also known as the Row/Column ID method), which constructs low-rank approximations using actual observed rows and columns from the original matrix. These methods offer superior alternatives for maintaining accuracy during low-rank matrix approximations in distributed settings.

Finally, we provide an empirical validation of our framework through extensive experimentation. Utilizing the WikiText dataset, we benchmark the convergence rates of our asynchronous framework against synchronous baselines. By comparing the empirical number of communication rounds required for convergence across different linear algebraic approximations, we identify the optimal decomposition strategy for distributed optimization in large-scale language modeling tasks.

2.3 Implementation and Evaluation

Specifically, we leverage fundamental linear algebraic operations and dimensionality reduction techniques, such as Principal Component Analysis (PCA), to streamline model updates and mitigate communication bottlenecks. By projecting high-dimensional gradients onto a lower-dimensional subspace, PCA extracts the most informative components, thereby significantly reducing the volume of data transmitted between edge devices and the central server while preserving essential directional information.

Regarding algorithmic implementation, our framework incorporates a suite of optimization paradigms, including Stochastic Gradient Descent (SGD) and its mini-batch variants. These methods are employed to iteratively minimize the global objective function, which aggregates local empirical risks across the distributed client base. The update mechanisms utilize optimized matrix-vector operations to ensure computational tractability and promote stable convergence of the global model.

To rigorously benchmark the performance of the proposed methods, we evaluate several key performance indicators (KPIs), including convergence rate, top-k accuracy, and communication-to-computation efficiency. Our experimental design encompasses diverse scenarios characterized by varying degrees of data heterogeneity (non-IIDness) and fluctuating device capabilities, ensuring the robustness and scalability of the framework under real-world constraints. Furthermore, we perform spectral analysis via eigenvalue decomposition of the Hessian matrix to investigate the local

curvature of the loss landscape, providing theoretical insights into the convergence dynamics of our asynchronous optimization paradigm.

Algorithm 1: Federated Learning with Gradient Simplification (PCA/QRCP/CUR)

Initialize: $w^{(0)}$, normalize data to $[-1, 1]$.
Given: number of clients n, learning rate η, threshold ε, max iterations T, local epochs k, rank r.
Method: $M \in \text{PCA}, \text{QRCP}, \text{CUR}\}$
for $t = 1 \to T$ **do**
  for $i = 1 \to n$ **(in parallel) do**
    $G_i^{(t)} = \nabla F(w^{(t-1)}, x_i)$
  end for
  $G^{(t)} = \frac{1}{n} \sum_{i=1}^{n} G_i^{(t)}$
  if $t = 1$ **then**
    $S_t = G^{(1)}$
  else
    $S_t = S_{t-1} + G^{(t)}$
  end if
  if $t \bmod k = 0$ **then** ▷ Dimension reduction step based on M
    $\bar{S}_t = S_t - \text{mean}(S_t, \text{axis} = 0)$ ▷ Center the data
    if $M = \text{PCA}$ **then** ▷ PCA via direct SVD on centered data
      $m =$ number of rows in $\bar{S}_t$
      $(U, S, V^\top) = \text{SVD}(\bar{S}_t,\ \text{full\_matrices} = \text{False})$
      $\text{explained\_variance} = \frac{S^2}{m-1}$
      ▷ Select top r components
      $V_r = V^\top[0 : r, :]^\top$ ▷ $V_r \in R^{d \times r}$
      ▷ Approximate S_t using the top r principal components
      $S_t^{\text{approx}} = (\bar{S}_t V_r) V_r^\top$
    else if $M = \text{QRCP}$ **then** ▷ QRCP on centered data
      $(Q, R, P) = \text{QRCP}(\bar{S}_t)$
      $S_t^{\text{approx}} = Q[:, 1 : r] R[1 : r, :] P^\top$
    else if $M = \text{CUR}$ **then** ▷ CUR decomposition on centered data
      $R_t = \text{SelectRows}(\bar{S}_t, r)$
      $Z_t = \text{FindZ}(\bar{S}_t, R_t)$
      $S_t^{\text{approx}} = Z_t R_t$
    end if
  else
    $S_t^{\text{approx}} = S_t$
  end if
  $w^{(t)} = w^{(t-1)} - \eta S_t^{\text{approx}}$
  $\Delta F = -S_t^{\text{approx}} \Delta x$ ▷ Δx is the step estimate
  if $\|\Delta F\| \le \varepsilon$ **then**
    break
  end if
end for

Definition 1 (Local-Global Objective Gap)**.** For the global optimum $\mathbf{w}^* = \arg\min_{\mathbf{w}} F(\mathbf{w})$ and local optimum $\mathbf{w}_k^* = \arg\min_{\mathbf{w}} F_k(\mathbf{w})$, we define the local-global objective gap as

$$\Gamma \triangleq F^* - \sum_{k=1}^{K} p_k F_k^* = \sum_{k=1}^{K} p_k \big(F_k(\mathbf{w}^*) - F_k(\mathbf{w}_k^*)\big) \ge 0. \tag{3}$$

This is the definition of Γ in section 3.

Definition 2 (Selection Skew)**.** For any $k \in S(\pi, \mathbf{w})$, we define

$$\rho(S(\pi, \mathbf{w}), \mathbf{w}') = \frac{\mathbb{E}_{S(\pi,\mathbf{w})} \left[\frac{1}{m} \sum_{k \in S(\pi,\mathbf{w})} \left(F_k(\mathbf{w}') - F_k^*\right)\right]}{F(\mathbf{w}') - \sum_{k=1}^{K} p_k F_k^*} \ge 0, \tag{4}$$

which reflects the skew of a client selection strategy π. The first $\mathbf{w}$ in $\rho(S(\pi, \mathbf{w}), \mathbf{w}')$ is the parameter vector that governs the client selection, and $\mathbf{w}'$ is the point at which F_k and F in the numerator and denominator, respectively, are evaluated. Note, $\mathbb{E}_{S(\pi,\mathbf{w})}[\cdot]$ is the expectation over the randomness from the selection strategy π, since there can be multiple sets S that π can map from a specific $\mathbf{w}$.

Since $\rho(S(\pi, \mathbf{w}), \mathbf{w}')$ is a function of versions of the global model $\mathbf{w}$ and $\mathbf{w}'$, which change during training, we define two related metrics that are independent of $\mathbf{w}$ and $\mathbf{w}'$. These metrics enable us to obtain a conservative error bound in the convergence analysis:

$$\bar{\rho} \triangleq \min_{\mathbf{w},\mathbf{w}'} \rho(S(\pi, \mathbf{w}), \mathbf{w}'), \quad \tilde{\rho} \triangleq \max_{\mathbf{w}} \rho(S(\pi, \mathbf{w}), \mathbf{w}^*), \tag{5}$$

where $\mathbf{w}^* = \arg\min_{\mathbf{w}} F(\mathbf{w})$. From equation 8, we have $\bar{\rho} \leq \tilde{\rho}$ for any client selection strategy π.

By integrating these techniques and metrics into our framework, we aim to achieve a more efficient and flexible federated learning system that is well-suited for large language models while addressing challenges like device heterogeneity and communication overhead.

After the derivation, we obtain the equation for time:

$$t = \log_\lambda \left(\frac{F\left(\overline{\mathbf{w}}^{(t)}\right) - F^* - x}{\frac{L}{\mu} A_1 - x} \right)$$

Let $F\left(\overline{\mathbf{w}}^{(0)}\right) - F^* = A_1$ and

$$\left[1 - n\mu\left(1 + \frac{3\bar{\rho}}{8}\right)\right] = \lambda,$$

and we have:

$$x = \frac{4L(\eta C + D)}{\mu(8 + 3\bar{\rho})},$$

where η is the learning rate, and $\bar{\rho}$ is the reflection of the skew of a client selection strategy π. Let L be the L-smoothness constant and $\mu = \frac{1}{L}$. Define:

$$C = 32\tau^2 G^2 + \sigma^2 + 6\rho L\Gamma, \quad D = 2\Gamma(\tilde{\rho} - \bar{\rho}).$$

The typical value of C is typically between 0.1 and 0.03, and the typical value of D is between 0 and 1 in practice.

We put the definitions of these parameters in Appendix.

We primarily aim to solve the equation below:

$$\frac{F\left(\mathbf{w}^{(t)}\right) - F\left(\mathbf{w}^{(t-1)}\right)}{x^t - x^{t-1}} = \nabla F^t(\mathbf{w}), \quad \text{where} \quad \begin{cases} \text{Let } \nabla F^t(\mathbf{w}) = G^t, \\ x^t - x^{t-1} = \Delta t. \end{cases}$$

We make a further assumption for an unknown T. As, the gradient is always descending. It must stop at some time step T and this is the local optimal point where we have $F^* = F(\mathbf{w}^T)$. We say the gradient descending algorithms is converged at T.

$F\left(\overline{\mathbf{w}}^{(t)}\right)$, F^* and $F(\bar{\mathbf{w}}^{(0)})$ are all unknown and we need to have some further mathematical calculation. We use $\langle\cdot\rangle$ to describe the averaged variable over all clients.

$$
\begin{aligned}
F(\mathbf{w}^{(t)}) - F(\mathbf{w}^{(t-1)}) &= G^t\,(x^t - x^{t-1}), \\
&= G^t\,\Delta x, \\
F(\mathbf{w}^{(t-1)}) - F(\mathbf{w}^{(t-2)}) &= G^{t-1}\,(\Delta x) \\
&\dots \\
F(\mathbf{w}^{1}) - F(\mathbf{w}^{0}) &= G^{1}\,(\Delta x)
\end{aligned}
$$

Thus, we know that at random step t,

$$
\begin{aligned}
F\left(\mathbf{w}^{(t)}\right) - F\left(\mathbf{w}^{(T)}\right) &= -(\,G^{t+1} + \ldots + G^{T}\,)(\Delta x), \\
F\left(\mathbf{w}^{0}\right) - F\left(\mathbf{w}^{(T)}\right) &= -(\,G^{1} + \ldots + G^{T}\,)(\Delta x),
\end{aligned}
$$

The expectation of Δx is easy to calculate which is the sensitivity of data. As we normalize our data on the range of $[-1, 1]$. We can get $\Delta = 4$. The key part to simplify the calculation of t is calculate G . Here is what we are going to do with G in our algorithms:

We initiate the federated learning process by executing local training on each participating client. In each iteration, we obtain the local gradients following backpropagation; for instance, in a setting with $n = 10$ clients, we retrieve the corresponding gradient matrices from each decentralized node.

Subsequently, we perform a global aggregation to compute the average gradient $G^t = \frac{1}{n}\sum_{i=1}^{n} G^i$. A primary challenge arises as G^t is typically a high-dimensional and complex matrix, posing significant computational burdens for downstream optimization. To address this, we introduce linear algebraic approximation techniques to simplify the required computations. Specifically, we utilize these linear approximation methods to derive the theoretical convergence bound t. Once the current communication round reaches the calculated threshold t, the training process is terminated under the assumption of convergence. We then empirically validate the final model accuracy and system latency to assess the fidelity of our approximation strategy.

3 Linear Approximation Techniques in Algorithm 1

To identify the most effective strategy for Algorithm 1, we systematically evaluate and integrate several matrix decomposition and approximation techniques. Each method offers distinct trade-offs between computational overhead, communication costs, and approximation error. The mathematical application of these methods within our framework is detailed below.

3.1 PCA

Let $\mathbf{X} \in \mathbb{R}^{n \times d}$ be a data matrix constructed by aggregating parameter arrays extracted from the model, where each row represents a parameter instance and each column corresponds to a feature dimension. To capture the dominant modes of variation within this parameter space, we perform Principal Component Analysis (PCA) as follows:

- **Data Centering:** Subtract the mean of each feature to center the data:

$$\tilde{\mathbf{X}} = \mathbf{X} - \frac{1}{n}\mathbf{1}\mathbf{1}^{\top}\mathbf{X},$$

  where $\mathbf{1}$ is the $n \times 1$ vector of all ones.

- **Singular Value Decomposition (SVD):** Perform SVD on the centered data $\tilde{\mathbf{X}}$:

$$\tilde{\mathbf{X}} = \mathbf{U}\mathbf{S}\mathbf{V}^{\top},$$

  where $\mathbf{U} \in \mathbb{R}^{n \times n}$, $\mathbf{V} \in \mathbb{R}^{d \times d}$ are orthonormal, and $\mathbf{S} = \mathrm{diag}(s_1, s_2, \ldots, s_{\min(n,d)})$ contains the singular values.

- **Variance Explained by Principal Components:** Each principal component (PC) corresponds to a singular value s_j. The variance explained by the j-th PC is:

$$\text{Var}(\text{PC}_j) = \frac{s_j^2}{n-1},$$

which matches the PCA formulation since the singular values squared are eigenvalues of the covariance matrix:

$$\mathbf{C} = \frac{1}{n-1}\tilde{\mathbf{X}}^\top \tilde{\mathbf{X}}.$$

By extracting the singular values and computing the explained variance as described, this procedure is mathematically equivalent to performing standard PCA on the parameter arrays. This allows for the identification of principal directions in the parameter space that capture maximal variance, facilitating high-dimensional dimensionality reduction, variance analysis, and subsequent interpretative downstream tasks.

3.2 QRCP Decomposition (Row-Based)

The QR Decomposition with Column Pivoting (QRCP) is employed to construct a rank-revealing approximation of the matrix S_t by identifying its most significant row structures. The decomposition factorizes the transposed matrix $S_t^\top$ as:

$$S_t^\top P = QR,$$

where:

- Q is an orthogonal matrix representing the basis vectors,
- R is an upper triangular matrix whose diagonal elements are non-increasing due to pivoting,
- P is a permutation matrix that encodes the row pivoting strategy.

To derive a low-rank approximation of S_t, we truncate the decomposition by retaining only the first k columns of Q and the first k rows of R:

$$G_{\text{approx},k} = (R^{(k)})^\top (Q^{(k)})^\top,$$

where:

- $Q^{(k)}$ denotes the truncated orthogonal matrix consisting of the first k columns of Q,
- $R^{(k)}$ denotes the first k rows of the upper triangular matrix R.

For the specific configuration of $k = 10$, the low-rank approximation is given by:

$$G_{\text{approx},10} \approx (R^{(10)})^\top (Q^{(10)})^\top.$$

This approach prioritizes the most informative rows of S_t through pivoting, ensuring numerical stability and preserving the fundamental geometric structure of the matrix even in rank-deficient scenarios.

The reconstructed approximation is formally expressed as:

$$S_{\text{approx}} = Q^{(k)} R^{(k)}.$$

Implementation Details

In practice, the QRCP decomposition is applied to the transposed parameter matrices following these steps:

1. Transpose the parameter matrix S_t to focus the decomposition on row-wise importance.
2. Execute the QRCP algorithm on $S_t^\top$, yielding the factors Q, R, and P.
3. Retain the top k components, where $\mathbf{k} = \min(\mathbf{10}, \text{rank}(\mathbf{R}))$.
4. Reconstruct the low-rank approximation S_{approx} using the reduced matrices $Q^{(k)}$ and $R^{(k)}$.

By leveraging row pivoting to select the most critical components, this decomposition provides a compact and numerically robust low-rank representation of S_t.

3.3 CUR Decomposition (Row ID Method)

The CUR decomposition facilitates a low-rank approximation by selecting actual subsets of rows and/or columns from the original matrix. In our Row Interpolative Decomposition (Row ID) approach, we approximate the matrix S_t using a representative subset of its rows and a computed interpolation matrix. This preservation of original row data is particularly advantageous for maintaining the interpretability of gradient updates. For a given parameter matrix S_t, the CUR Row ID method proceeds as follows:

1. **Row Selection:** We select k representative rows from S_t to form the submatrix R_t. These rows are sampled without replacement to ensure a diverse representation of the parameter manifold.
2. **Interpolation Matrix Computation:** We determine the interpolation matrix Z_t that minimizes the Frobenius norm of the reconstruction error:

$$Z_t = \arg\min_Z \|S_t - ZR_t\|_F^2.$$

   This is solved as a least-squares problem:

$$Z_t^\top = (R_t^\dagger)^\top S_t^\top,$$

   where $R_t^\dagger$ denotes the Moore-Penrose pseudoinverse of R_t.
3. **Reconstruction:** The approximation of the original matrix is then reconstructed using R_t and Z_t:

$$S_{\text{approx},t} = Z_t R_t.$$

   The resulting matrix $S_{\text{approx},t}$ is then reshaped to match the original dimensions of S_t.

Implementation Details

The Row ID method is integrated into our framework by operating on the flattened parameter matrices:

- Each parameter tensor is reshaped into a 2D matrix, where the first dimension corresponds to the row space.
- The rank parameter k is dynamically set as $\mathbf{k} = \min(\mathbf{10}, \text{number of rows in the parameter})$.
- Row selection is performed via random sampling without replacement to maintain data integrity.
- The interpolation matrix Z_t is computed using an efficient least-squares formulation.
- Finally, the reconstructed matrix $S_{\text{approx},t}$ is reshaped back to the original tensor dimensions for model updating.

Error Analysis

To evaluate the quality of the reconstruction, the Frobenius norm of the difference between the original matrix S_t and its approximation $S_{\text{approx},t}$ is computed:

$$\text{Reconstruction Error} = \|S_t - S_{\text{approx},t}\|_F.$$

The mean reconstruction error over all model parameters is also calculated and logged:

$$\text{Mean Reconstruction Error} = \frac{1}{n}\sum_{i=1}^{n} \|S_{t,i} - S_{\text{approx},t,i}\|_F,$$

where n is the number of parameter matrices.

Summary

The CUR decomposition with Row ID provides a computationally efficient way to approximate large model parameter matrices while preserving their most important structural information. By incorporating reconstruction error into the stopping condition, this method ensures both accuracy and resource efficiency during the training process.

3.4 Stopping Condition

The mean reconstruction error, denoted as δ_g, is used as part of the stopping condition for the training process. Based on predefined constants n, μ, $\bar{\rho}$, and L, the parameter λ is calculated:

$$\lambda = 1 - n\mu\left(1 + \frac{3\bar{\rho}}{8}\right).$$

Subsequently, the stopping criterion is defined as:

$$t = \log_\lambda\left(\frac{\delta_g - x}{\frac{L}{\mu}A_1 - x}\right),$$

where x is a function of the model-specific constants L, C, and D. If $t \geq$ current round, the training process halts.

Each of these methods offers unique trade-offs between computational complexity, communication efficiency, and approximation accuracy. The choice of method should be guided by the characteristics of the gradient matrix S_t and the specific requirements of the federated learning system.

4 Experiment

In this study, we fine-tune a GPT-2 model using the Wikitext dataset[1], which contains 18,666 entries. The task is text generation, where the model predicts the next word for each sentence, ensuring it can generate content similar to the Wikitext data.

Federated Learning Setup: We conducted training in a federated learning (FL) environment with two clients. In each communication round:

1. The server distributes the current global model parameters to both n clients where we set $n = 2$
2. Each client locally fine-tunes the model for one epoch over its assigned portion of the dataset.
3. Clients send their locally updated parameters back to the server.
4. The server aggregates these updates (e.g., via Federated Averaging) into *updated_weights*, forming a new global model state that integrates the knowledge from all clients without sharing raw data.

Table 1: Representative Parameter Matrices and Their Dimensions in the Aggregated Model

Parameter	Dimensions	Layer Type	Description
Token Embedding	$(50257, 768)$	Embedding Layer	Maps a large vocabulary to a dense embedding space. Each token is represented by a 768-dimensional vector where we have 50257 tokens at maximal.
Positional Embedding	$(1024, 768)$	Embedding Layer	Encodes sequence position information up to length 1024, assigning a unique embedding to each position in the sequence which can be extended to 50257.
Attention Weights	$(768, 2304)$	Multi-Head Attention	A projection matrix used for query, key, and value transformations in self-attention, expanding the feature space from 768 to 2304 dimensions.
Feed-For Network	$(3072, 768)$	Transformer Block	A reduction step after the feed-for expansion. The feed-for layer typically first expands the dimension from 768 to a higher dimension (e.g., 3072), then maps it back down to 768 for stable and expressive representations.

Model Parameter Shapes in GPT-2: The dimensions presented in Table 1 highlight the high-dimensional and heterogeneous nature of parameters within the GPT-2 model. Large vocabulary sizes and extended sequence lengths necessitate extensive embedding matrices, while the transformer layers introduce parameter matrices that undergo expansion and compression, resulting in diverse shapes.

These variations in parameter dimensions directly influence the feasibility and efficiency of subsequent matrix decompositions, such as PCA, QRCP, or CUR. Larger matrices, although more challenging to factorize, may offer richer low-rank structures that can be leveraged to reduce communication overhead. Meanwhile, structural differences across parameter sets may require specialized or adaptive decomposition methods to ensure both computational efficiency and stable performance. Accordingly, the following experiments will focus on selecting and refining these decomposition approaches to improve training speed and scalability without substantially compromising model accuracy.

In GPT-2, model parameters at iteration t are represented by $\mathbf{w}^{(t)}$, and the updates to these parameters are stored in `updated_weights`, keyed by layer names. The difference in the model's loss across iterations, $F(\mathbf{w}^{(t)}) - F(\mathbf{w}^{(t-1)})$, highlights the impact of these updates. By applying linear algebra approximations (e.g., PCA) to the tensors in `updated_weights`, we can reveal dominant variance directions after flattening them into matrices. This analysis fosters a deeper understanding of GPT-2's parameter evolution during federated training, aiding in the comparison of different distributed optimization methods and ultimately providing enhanced insights.

Dimensionality Reduction Techniques (PCA, QRCP, CUR): To improve efficiency and reduce complexity, we integrate dimensionality reduction and matrix decomposition techniques—Principal Component Analysis (PCA), QR with Column Pivoting (QRCP), and CUR decomposition—into the federated learning process. These methods can influence:

- **Computational Efficiency:** Reducing parameter dimensions may speed up training or aggregation.
- **Convergence Behavior:** Certain decompositions might stabilize training or lead to faster convergence.
- **Accuracy (Perplexity):** Compression or transformation may affect the model's predictive performance.

Evaluation Metrics: We measure performance using:

[1] `https://huggingface.co/datasets/Salesforce/wikitext`

Table 2: Result Training Metrics for PCA

Train Runtime (s)	Samples per Second	Steps per Second	Train Loss	Epoch
5.6238	3319.083	0.178	4.4335	1.0
6.7765	2754.527	0.148	4.3179	1.0
6.9538	2684.285	0.144	4.1241	1.0
7.0656	2641.832	0.142	4.0038	1.0
6.8458	2726.647	0.146	3.7639	1.0
7.6762	2431.666	0.13	3.8856	1.0
6.7615	2760.648	0.148	3.6764	1.0
5.8637	3183.32	0.171	3.5578	1.0
6.9611	2681.457	0.144	3.4772	1.0

1. **Elapsed Time:** The total wall-clock time until convergence. Lower latency indicates more efficient training.
2. **Perplexity:** A standard metric for language models that gauges their predictive capability. Lower perplexity means the model is better at predicting the next word, indicating more accurate language modeling.

We run the federated training for five total rounds to observe trends over time, and repeat each method six times, averaging the results. This approach helps us understand how each technique affects training speed, convergence quality, and accuracy.

Context in our Federated Learning procedure:

Local Training: Each client trains the GPT-2 model on their assigned subset of the Wikitext dataset (with a total size of approximately 18,666 samples).

Updates to the Server: Clients send back their locally trained model updates (weights) without sharing any raw data.

Aggregation: The server performs federated averaging on these updates, producing *updated_weights*. These aggregated weights integrate the learned representations from all clients, thereby improving the global model performance.

Matrix Size: During each communication round of federated training, the model parameters undergo local updates on participating clients and subsequent aggregation on the server. These updates produce large parameter tensors whose dimensions play a critical role in selecting efficient matrix decomposition methods.

The *updated_weights* represent the newly aggregated model parameters after each federated learning round, reflecting the combined improvements from multiple clients. The parameter shapes correspond to GPT-2's embeddings and other model components, each tied to its specific role in the network (tokens, positions, or bias/normalization). By incorporating PCA, QRCP, and CUR techniques and comparing their effects on latency and perplexity, we gain valuable insights into optimizing GPT-2-based models for distributed training environments.

Table 3: Result Training Metrics for QRCP

Train Runtime (s)	**Samples per Second**	**Steps per Second**	**Train Loss**	**Epoch**
6.7466	2766.721	0.148	4.3179	1.0
7.6349	2444.825	0.131	4.4335	1.0
7.1882	2596.754	0.139	4.0038	1.0
6.8067	2742.282	0.147	4.1241	1.0
7.4834	2494.309	0.134	3.7639	1.0
7.5199	2482.198	0.133	3.8856	1.0
7.6213	2449.192	0.131	3.5578	1.0
7.2237	2584.011	0.138	3.6764	1.0
5.8933	3167.318	0.170	3.3598	1.0
7.3777	2530.052	0.136	3.4772	1.0

Table 4: Averaged PCA results

Round	Time Elapsed (s)	Accuracy (%)
1	114.42	54.38
2	241.62	51.34
3	369.41	49.80
4	496.81	48.94
5	623.72	48.55

Table 5: Averaged QRCP Results

Round	Elapsed Time	Accuracy
1	354.658	54.380
2	629.019	51.341
3	939.959	49.803
4	1265.268	48.936
5	1578.228	48.551

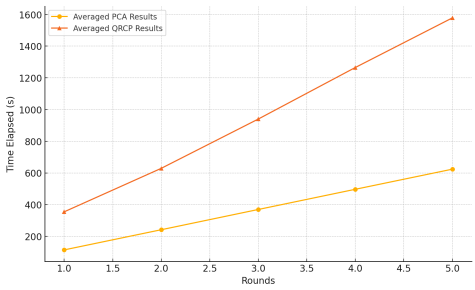

Figure 1: Comparison of PCA and QRCP: Elapsed Time vs. Accuracy Across Training Rounds

Figure 2: Averaged CUR Results, change to figure

round	elapsed time	accuracy
1	242.0706	7944.818
2	567.0051	7043.070

Based on the empirical results summarized in the table, the CUR decomposition exhibits the most aggressive convergence behavior, reaching the stopping criterion in only two communication rounds, whereas PCA and QRCP require five rounds. However, this accelerated termination comes at a notable cost to predictive fidelity, as CUR yields significantly lower accuracy compared to its counterparts. This performance gap suggests that while CUR effectively captures structural sketches, it may discard fine-grained gradient information essential for LLM fine-tuning. To ensure statistical rigor, all metrics were averaged over six independent trials. Among the evaluated techniques, PCA demonstrated the highest computational efficiency in terms of average wall-clock latency. Furthermore, the comparable accuracy achieved by both PCA and QRCP indicates a Pareto-optimal trade-off between execution time and model precision. Future research should focus on refining the row-selection heuristics in CUR to enhance its representational accuracy while preserving its superior convergence speed, particularly for latency-sensitive or large-scale distributed tasks.

5 Future Work

Several research avenues warrant further investigation to extend the capabilities of the proposed framework. First, we plan to develop adaptive client selection strategies that employ online heuris-

tics to dynamically respond to fluctuating data distributions and transient hardware constraints. Second, we intend to explore the integration of Reinforcement Learning (RL) to optimize the selection of matrix approximation ranks and hyperparameters, enabling the system to autonomously learn optimal configurations through environmental interactions. Third, a critical frontier remains the scalability of our matrix approximation techniques when applied to ultra-large-scale models exceeding billions of parameters. Addressing these challenges will further enhance the framework's robustness and generalizability in complex, non-stationary real-world environments.

6 Conclusion

The proposed fully asynchronous federated learning framework represents a significant advancement in distributed machine learning, specifically optimized for the unique challenges of Large Language Models. By decoupling local updates from global synchronization, the framework effectively eliminates straggler-induced bottlenecks and accelerates convergence in the presence of extreme device heterogeneity. The integration of targeted data selection and dynamic optimization ensures that the system remains both personalized to individual users and robust at the global scale. Ultimately, this research contributes a theoretically grounded and practically viable architecture for private, large-scale distributed learning, and fosters innovation across privacy-sensitive domains in mobile and IoT ecosystems.

References

Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459, 2010.

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Thomas Bäck and Hans-Paul Schwefel. An overview of evolutionary algorithms for parameter optimization. *Evolutionary computation*, 1(1):1–23, 1993.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.

Tom B Brown. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

Yae Jee Cho, Jianyu Wang, and Gauri Joshi. Client selection in federated learning: Convergence analysis and power-of-choice selection strategies. *arXiv preprint arXiv:2010.01243*, 2020.

Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and trends® in machine learning*, 14(1–2):1–210, 2021.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pp. 1273–1282. PMLR, 2017.

Alec Radford. Improving language understanding by generative pre-training. 2018.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Jianwei Xiao, Ming Gu, and Julien Langou. Fast parallel randomized qr with column pivoting algorithms for reliable low-rank matrix approximations. *2017 IEEE 24th International Conference on High Performance Computing (HiPC)*, pp. 233–242, 2017. URL https://api.semanticscholar.org/CorpusID:3444436.

A Derivation of Convergence Bounds

Consider a cross-device federated learning setup with total K clients, where client k has a local dataset $\mathcal{B}_k$ consisting $|\mathcal{B}_k| = D_k$ data samples. The clients are connected via a central aggregating server, and seek to collectively find the model parameter $\mathbf{w}$ that minimizes the empirical risk:

$$F(\mathbf{w}) = \frac{1}{\sum_{k=1}^{K} D_k} \sum_{k=1}^{K} \sum_{\xi \in \mathcal{B}_k} f(\mathbf{w}, \xi) = \sum_{k=1}^{K} p_k F_k(\mathbf{w})$$

where $f(\mathbf{w}, \xi)$ is the composite loss function for sample ξ and parameter vector $\mathbf{w}$. The term $p_k = D_k / \sum_{k=1}^{K} D_k$ is the fraction of data at the k-th client, and $F_k(\mathbf{w}) = \frac{1}{|\mathcal{B}_k|} \sum_{\xi \in \mathcal{B}_k} f(\mathbf{w}, \xi)$ is the local objective function of client k. In federated learning, the vectors $\mathbf{w}^*$ and $\mathbf{w}_k^*$ for $k = 1, \ldots, K$ that minimize $F(\mathbf{w})$ and $F_k(\mathbf{w})$ respectively can be very different from each other. We define $F^* = \min_{\mathbf{w}} F(\mathbf{w}) = F(\mathbf{w}^*)$ and $F_k^* = \min_{\mathbf{w}} F_k(\mathbf{w}) = F_k(\mathbf{w}_k^*)$.

Formally, we index the local SGD iterations with $t \geq 0$. The set of active clients at iteration t is denoted by $\mathcal{S}^{(t)}$. Since active clients perform τ steps of local update, the active set $\mathcal{S}^{(t)}$ also remains constant for every τ iterations. That is, if $(t+1) \mod \tau = 0$, then $\mathcal{S}^{(t+1)} = \mathcal{S}^{(t+2)} = \cdots = \mathcal{S}^{(t+\tau)}$. Accordingly, the update rule of FedAvg can be written as follows:

$$\mathbf{w}_k^{(t+1)} = \begin{cases} \mathbf{w}_k^{(t)} - \eta_t g_k\left(\mathbf{w}_k^{(t)}, \xi_k^{(t)}\right), & \text{for } (t+1) \mod \tau \neq 0, \\ \frac{1}{m} \sum_{j \in \mathcal{S}^{(t)}} \left(\mathbf{w}_j^{(t)} - \eta_t g_j\left(\mathbf{w}_j^{(t)}, \xi_j^{(t)}\right)\right) \triangleq \overline{\mathbf{w}}^{(t+1)}, & \text{for } (t+1) \mod \tau = 0, \end{cases} \tag{1}$$

where $\mathbf{w}_k^{(t+1)}$ denotes the local model parameters of client k at iteration t, η_t is the learning rate, and

$$g_k\left(\mathbf{w}_k^{(t)}, \xi_k^{(t)}\right) = \frac{1}{b} \sum_{\xi \in \xi_k^{(t)}} \nabla f\left(\mathbf{w}_k^{(t)}, \xi\right)$$

is the stochastic gradient over mini-batch $\xi_k^{(t)}$ of size b that is randomly sampled from client k's local dataset $\mathcal{B}_k$.

We have the stochastic gradient's expected squared norm is uniformly bounded,

$i.e., \mathbb{E}\|\mathbf{g}_k(\mathbf{w}_k, \xi_k)\|^2 \leq G^2 \quad \text{for } k = 1, \ldots, K.$

This is the definition of $\mathbf{G}$ in section 3.

Definition 3 (Local-Global Objective Gap)**.** For the global optimum $\mathbf{w}^* = \arg\min_{\mathbf{w}} F(\mathbf{w})$ and local optimum $\mathbf{w}_k^* = \arg\min_{\mathbf{w}} F_k(\mathbf{w})$, we define the local-global objective gap as

$$\Gamma \triangleq F^* - \sum_{k=1}^{K} p_k F_k^* = \sum_{k=1}^{K} p_k \big(F_k(\mathbf{w}^*) - F_k(\mathbf{w}_k^*)\big) \geq 0. \tag{6}$$

This is the definition of Γ in section 3.

Definition 4 (Selection Skew)**.** For any $k \in S(\pi, \mathbf{w})$, we define

$$\rho(S(\pi, \mathbf{w}), \mathbf{w}') = \frac{\mathbb{E}_{S(\pi,\mathbf{w})}\left[\frac{1}{m} \sum_{k \in S(\pi,\mathbf{w})} \left(F_k(\mathbf{w}') - F_k^*\right)\right]}{F(\mathbf{w}') - \sum_{k=1}^{K} p_k F_k^*} \geq 0, \tag{7}$$

which reflects the skew of a client selection strategy π. The first $\mathbf{w}$ in $\rho(S(\pi, \mathbf{w}), \mathbf{w}')$ is the parameter vector that governs the client selection, and $\mathbf{w}'$ is the point at which F_k and F in the numerator and denominator, respectively, are evaluated. Note, $\mathbb{E}_{S(\pi,\mathbf{w})}[\cdot]$ is the expectation over the randomness from the selection strategy π, since there can be multiple sets S that π can map from a specific $\mathbf{w}$.

Since $\rho(S(\pi, \mathbf{w}), \mathbf{w}')$ is a function of versions of the global model $\mathbf{w}$ and $\mathbf{w}'$, which change during training, we define two related metrics that are independent of $\mathbf{w}$ and $\mathbf{w}'$. These metrics enable us to obtain a conservative error bound in the convergence analysis:

$$\bar{\rho} \triangleq \min_{\mathbf{w}, \mathbf{w}'} \rho(S(\pi, \mathbf{w}), \mathbf{w}'), \quad \tilde{\rho} \triangleq \max_{\mathbf{w}} \rho(S(\pi, \mathbf{w}), \mathbf{w}^*), \tag{8}$$

where $\mathbf{w}^* = \arg\min_{\mathbf{w}} F(\mathbf{w})$. From equation 8, we have $\bar{\rho} \leq \tilde{\rho}$ for any client selection strategy π.

Effect of the Client Selection Strategy on $\bar{\rho}$ and $\tilde{\rho}$. For the unbiased client selection strategy π_{rand}, we have $\rho(S(\pi_{\text{rand}}, \mathbf{w}), \mathbf{w}') = 1$ for all $\mathbf{w}$ and $\mathbf{w}'$ since the numerator and denominator of equation 7 become equal, and $\bar{\rho} = \tilde{\rho} = 1$. For a client selection strategy π that chooses clients with higher $F_k(\mathbf{w})$ more often, $\bar{\rho}$ and $\tilde{\rho}$ will be larger (and ≥ 1). In the convergence analysis, we show that a larger $\bar{\rho}$ implies faster convergence, albeit with a potential error gap, which is proportional to $(\tilde{\rho}/\bar{\rho} - 1)$. Motivated by this, in Section 4 we present an adaptive client selection strategy that prefers selecting clients with higher loss $F_k(\mathbf{w})$ and achieves faster convergence speed with low solution bias.

Moreover, $\overline{\mathbf{w}}^{(t+1)}$ denotes the global model at server. Although $\overline{\mathbf{w}}^{(t)}$ is only updated after every τ iterations, for the purpose of convergence analysis we consider a virtual sequence of $\overline{\mathbf{w}}^{(t)}$ that is updated at each iteration as follows:

$$\overline{\mathbf{w}}^{(t+1)} = \overline{\mathbf{w}}^{(t)} - \eta_t \overline{g}^{(t)} = \overline{\mathbf{w}}^{(t)} - \eta_t \left(\frac{1}{m} \sum_{k \in \mathcal{S}^{(t)}} g_k \left(\mathbf{w}_k^{(t)}, \xi_k^{(t)} \right) \right) \tag{2}$$

with $\overline{g}^{(t)} = \frac{1}{m} \sum_{k \in \mathcal{S}^{(t)}} g_k \left(\mathbf{w}_k^{(t)}, \xi_k^{(t)} \right)$.

Note that in equations (1) and (2), we do not weight the client models by their dataset fractions p_k because p_k is considered in the client selection scheme used to decide the set $\mathcal{S}^{(t)}$.

$$\begin{aligned}
\mathbb{E}[\|\overline{\mathbf{w}}^{(t)} - \mathbf{w}^*\|^2] &= \mathbb{E}[\|\frac{1}{m} \sum_{k \in \mathcal{S}^{(t)}} \mathbf{w}_k^{(t)} - \mathbf{w}^*\|^2] = \mathbb{E}[\|\frac{1}{m} \sum_{k \in \mathcal{S}^{(t)}} \mathbf{w}_k^{(t)} - \mathbf{w}^*\|^2] \\
&\leq \frac{1}{m} \mathbb{E}[\sum_{k \in \mathcal{S}^{(t)}} \|\mathbf{w}_k^{(t)} - \mathbf{w}^*\|^2]
\end{aligned}$$

With $\overline{g}^{(t)} = \frac{1}{m} \sum_{k \in \mathcal{S}^{(t)}} g_k \left(\mathbf{w}_k^{(t)}, \xi_k^{(t)} \right)$ as defined above, we have that

$$\begin{aligned}
\|\overline{\mathbf{w}}^{(t+1)} - \mathbf{w}^*\|^2 &= \|\overline{\mathbf{w}}^{(t)} - \eta_t \overline{\mathbf{g}}^{(t)} - \mathbf{w}^*\|^2 \\
&= \|\overline{\mathbf{w}}^{(t)} - \mathbf{w}^* - \frac{\eta_t}{m} \sum_{k \in \mathcal{S}^{(t)}} \nabla F_k \left(\mathbf{w}_k^{(t)} \right) \|^2 + \frac{\eta_t^2}{m} \sum_{k \in \mathcal{S}^{(t)}} \|\nabla F_k \left(\mathbf{w}_k^{(t)} \right) \|^2. \\
&= \underbrace{\|\mathbf{w}^{(t)} - \mathbf{w}^*\|^2 - 2\eta_t \langle \mathbf{w}^{(t)} - \mathbf{w}^*, \frac{1}{m} \sum_{k \in S(t)} \nabla F_k(\mathbf{w}_k^{(t)}) \rangle}_{A_1} \\
&\quad \underbrace{-2\eta_t \langle \mathbf{w}^{(t)} - \mathbf{w}^*, \frac{1}{m} \sum_{k \in S(t)} (\nabla F_k(\mathbf{w}_k^{(t)}) - \bar{\mathbf{g}}^{(t)}) \rangle}_{A_2} \\
&\quad \underbrace{+ \eta_t^2 \left\| \frac{1}{m} \sum_{k \in S(t)} \nabla F_k(\mathbf{w}_k^{(t)}) \right\|^2}_{A_3} \underbrace{+ \eta_t^2 \left\| \frac{1}{m} \sum_{k \in S(t)} (\nabla F_k(\mathbf{w}_k^{(t)}) - \bar{\mathbf{g}}^{(t)}) \right\|^2}_{A_4}.
\end{aligned}$$

First, let's bound A_1:

$$\begin{aligned}
&- 2\eta_t \langle \overline{\mathbf{w}}^{(t)} - \mathbf{w}^*, \frac{1}{m} \sum_{k \in S(t)} \nabla F_k(\mathbf{w}_k^{(t)}) \rangle = -\frac{2\eta_t}{m} \sum_{k \in S(t)} \langle \overline{\mathbf{w}}^{(t)} - \mathbf{w}^*, \nabla F_k(\mathbf{w}_k^{(t)}) \rangle \\
&= -\frac{2\eta_t}{m} \sum_{k \in S(t)} \langle \overline{\mathbf{w}}^{(t)} - \mathbf{w}_k^{(t)}, \nabla F_k(\mathbf{w}_k^{(t)}) \rangle - \frac{2\eta_t}{m} \sum_{k \in S(t)} \langle \mathbf{w}_k^{(t)} - \mathbf{w}^*, \nabla F_k(\mathbf{w}_k^{(t)}) \rangle \\
&\leq 16\eta_t^2 \tau^2 G^2 - \frac{\eta_t \mu}{m} \sum_{k \in S(t)} \|\mathbf{w}_k^{(t)} - \mathbf{w}^*\|^2 + \frac{2L\eta_t^2}{m} \sum_{k \in S(t)} \left(F_k(\mathbf{w}_k^{(t)}) - F_k^* \right) - \frac{2\eta_t}{m} \sum_{k \in S(t)} \left(F_k(\mathbf{w}_k^{(t)}) - F_k(\mathbf{w}^*) \right)
\end{aligned}$$

Next, in expectation, $\mathbb{E}[A_2] = 0$ due to the unbiased gradient.

We bound A_3 as follows:

$$\eta_t^2 \| \frac{1}{m} \sum_{k \in S(t)} \nabla F_k(\mathbf{w}_k^{(t)}) \|^2 = \frac{\eta_t^2}{m} \sum_{k \in S(t)} \| \nabla F_k(\mathbf{w}_k^{(t)}) \|^2 \leq \frac{2L\eta_t^2}{m} \sum_{k \in S(t)} \left(F_k(\mathbf{w}_k^{(t)}) - F_k^* \right)$$

Lastly we can bound A_4 using the bound of variance of stochastic gradients as

$$\begin{aligned}
\mathbb{E}\left[\eta_t^2 \| \frac{1}{m} \sum_{k \in S(t)} \nabla F_k(\mathbf{w}_k^{(t)}) - \mathbf{g}^{(t)} \|^2 \right] &= \eta_t^2 \mathbb{E}[\| \frac{1}{m} \sum_{k \in S(t)} (g_k(\mathbf{w}_k^{(t)}, \xi_k^{(t)}) - \nabla F_k(\mathbf{w}_k^{(t)})) \|^2] \\
&= \frac{\eta_t^2}{m^2} \mathbb{E}_{S(t)}[\sum_{k \in S(t)} \mathbb{E} \Big\| g_k(\mathbf{w}_k^{(t)}, \xi_k^{(t)}) - \nabla F_k(\mathbf{w}_k^{(t)}) \|^2] \\
&\leq \frac{\eta_t^2 \sigma^2}{m}.
\end{aligned}$$

Using the bounds of A_1, A_2, A_3, A_4 above we have

$$\begin{aligned}
\mathbb{E}[\|\mathbf{w}^{(t+1)} - \mathbf{w}^*\|^2] &\leq \mathbb{E}[\|\mathbf{w}^{(t)} - \mathbf{w}^*\|^2] - \frac{\eta_t \mu}{m} \mathbb{E}[\sum_{k \in S(t)} \|\mathbf{w}_k^{(t)} - \mathbf{w}^*\|^2] + 16\eta_t^2 \tau^2 G^2 \\
&\quad + \frac{\eta_t^2 \sigma^2}{m} + \frac{4L\eta_t^2}{m} \mathbb{E}[\sum_{k \in S(t)} (F_k(\mathbf{w}_k^{(t)}) - F_k^*)] - \frac{2\eta_t}{m} \mathbb{E}[\sum_{k \in S(t)} (F_k(\mathbf{w}_k^{(t)}) - F_k(\mathbf{w}^*))] \\
&\leq (1 - \eta_t \mu) \mathbb{E}[\|\mathbf{w}^{(t)} - \mathbf{w}^*\|^2] + 16\eta_t^2 \tau^2 G^2 + \frac{\eta_t^2 \sigma^2}{m} \\
&\quad \underbrace{+ \frac{4L\eta_t^2}{m} \mathbb{E}[\sum_{k \in S(t)} (F_k(\mathbf{w}_k^{(t)}) - F_k^*)] - \frac{2\eta_t}{m} \mathbb{E}[\sum_{k \in S(t)} (F_k(\mathbf{w}_k^{(t)}) - F_k(\mathbf{w}^*))]}_{A_5}
\end{aligned}$$

$$A_5 = \mathbb{E}[\underbrace{\frac{2\eta_t(2L\eta_t - 1)}{m} \sum_{k \in S(t)} \left(F_k(\mathbf{w}_k^{(t)}) - F_k^* \right)}_{A_6}] + 2\eta_t \mathbb{E}[\frac{1}{m} \sum_{k \in S(t)} \left(F_k(\mathbf{w}^*) - F_k^* \right)]$$

With $\eta_t < 1/4L$ and $\nu_t = 2\eta_t(1 - 2L\eta_t)$, we have that A_6 can be rewritten and bounded as

$$\begin{aligned}
&- \frac{\nu_t}{m} \sum_{k \in S(t)} \left(F_k(\mathbf{w}_k^{(t)}) - F_k(\bar{\mathbf{w}}^{(t)}) + F_k(\bar{\mathbf{w}}^{(t)}) - F_k^* \right) \\
&\leq - \frac{\nu_t}{m} (1 - \eta_t L) \sum_{k \in S(t)} \left(F_k(\bar{\mathbf{w}}^{(t)}) - F_k^* \right) + \frac{1}{m} \sum_{k \in S(t)} \|\mathbf{w}_k^{(t)} - \bar{\mathbf{w}}^{(t)}\|^2
\end{aligned}$$

Using $\Delta_t \leq \frac{2}{\mu} \left(F\left(\overline{\mathbf{w}}^{(t)} \right) - F^* \right)$.we have:

$$
\begin{aligned}
F\left(\overline{\mathbf{w}}^{(t)}\right) - F^* &\leq \frac{L}{\mu}(1-\eta\mu B)^t \left(F\left(\overline{\mathbf{w}}^{(0)}\right) - F^*\right) \\
&\quad + \frac{L(\eta C + D)}{2\mu B}\left(1 - (1-\eta\mu B)^t\right) \\
&= \frac{L}{\mu}\left[1 - \eta\mu\left(1 + \frac{3\bar{\rho}}{8}\right)\right]^t \left(F\left(\overline{\mathbf{w}}^{(0)}\right) - F^*\right) \\
&\quad + \frac{4L(\eta C + D)}{\mu(8+3\bar{\rho})}\left[1 - \left[1 - \eta\mu\left(1 + \frac{3\bar{\rho}}{8}\right)\right]^t\right].
\end{aligned}
$$

Let $F\left(\overline{\mathbf{w}}^{(0)}\right) - F^* = A_1$ and

$$
\left[1 - n\mu\left(1 + \frac{3\bar{\rho}}{8}\right)\right] = \lambda,
$$

and we have:

$$
\begin{aligned}
\frac{4L(\eta C + D)}{\mu(8+3\bar{\rho})} &= x, \\
\Rightarrow \frac{L}{\mu}\lambda^t A_1 + x\left(1 - \lambda^t\right) &= F\left(\overline{\mathbf{w}}^{(t)}\right) - F^*, \\
\Rightarrow \frac{L}{\mu}\lambda^t A_1 + x - \lambda^t &= F\left(\overline{\mathbf{w}}^{(t)}\right) - F^*, \\
\Rightarrow \frac{L}{\mu}\lambda^t A_1 - \lambda^t &= F\left(\overline{\mathbf{w}}^{(t)}\right) - F^* - x, \\
\Rightarrow \lambda^t\left[\frac{L}{\mu}A_1 - 1\right] &= F\left(\overline{\mathbf{w}}^{(t)}\right) - F^* - x, \\
\Rightarrow \lambda^t &= \frac{F\left(\overline{\mathbf{w}}^{(t)}\right) - F^* - x}{\frac{L}{\mu}A_1 - 1}, \\
\Rightarrow t &= \log_\lambda\left(\frac{F\left(\overline{\mathbf{w}}^{(t)}\right) - F^* - x}{\frac{L}{\mu}A_1 - x}\right).
\end{aligned}
$$

For $t = \log\frac{F(\overline{\mathbf{y}},t) - F^* - x}{\frac{L}{\mu}A_1 - 1}$ at each timestep t, we have that:

$$
\frac{F\left(\mathbf{w}^{(t)}\right) - F\left(\mathbf{w}^{(t-1)}\right)}{x^t - x^{t-1}} = \nabla F^t(\mathbf{w}) \quad \text{where} \quad \begin{cases} \text{let } \nabla F^t(\mathbf{w}) = G^t, \\ x^t - x^{t-1} = \Delta t. \end{cases}
$$