# REVISITING DATA SHARDING FOR DISTRIBUTED TRAINING WITH PERIODIC SYNCHRONIZATION

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

In datacenter distributed training, periodic model synchronization can mitigate communication bottlenecks and accelerate training. However, longer synchronization intervals exacerbate convergence slowdowns induced by data heterogeneity across nodes. To improve convergence efficiency without increasing model synchronization, we revisit data sharding and propose a weighted stratified sharding algorithm. Our algorithm exploits objective values observed during training to construct weighted shards that reduce objective discrepancies across nodes. We establish theoretical guarantees for the induced weighted objectives and derive a convergence bound in terms of the objective approximation error. The experiments span various distributed setups and training tasks, from image classification to math continual pretraining. Compared with widely used random partitioning, our sharding algorithm introduces negligible runtime overhead and converges up to 1.45 times faster in training steps. It achieves both lower training loss and improved test accuracy, and these gains persist even when using nearly 80% fewer synchronization rounds.

## 1 INTRODUCTION

Training large-scale machine learning models requires massive distributed computing resources and huge amount of data (Hoffmann et al., 2022). Data parallelism is a standard paradigm that enables these computing resources to process different data shards in parallel, compute gradients on their respective shards, and synchronize model updates (Shazeer et al., 2018; Li et al., 2020). However, modern training clusters contain thousands of GPUs within a single datacenter, or span multiple datacenters with constrained communication bandwidth (Jiang et al., 2024; Zhu et al., 2025). As model sizes and GPU counts continue to grow, **communication becomes an increasingly severe bottleneck**. In Meta's production clusters, for example, it can account for up to 60% of a training iteration (Wang et al., 2023). This overhead makes it challenging for data parallel training to synchronize models at every step (Dong et al., 2025).

An active line of work keeps the basic random sharding strategy of data parallelism and reduces communication through local updates with periodic synchronization (Koloskova et al., 2020; Wang & Joshi, 2021). A standard example is Local SGD (Stich, 2019) with periodic averaging, and many of its variants can be viewed through the FedOpt framework (Reddi et al., 2021; Cho et al., 2022). In datacenter distributed training, these ideas are adopted mainly for scalability and communication efficiency rather than data privacy (Douillard et al., 2024; Jaghouar & Hagemann, 2024; Sani et al., 2024; Jaghouar et al., 2024; Naganuma et al., 2025). Recent variants (Condat et al., 2025; Thérien et al., 2025; Nedelkoski et al., 2025; Douillard et al., 2025) modify local optimization or update aggregation to mitigate discrepancies among local objectives (Woodworth et al., 2020b; Patel et al., 2024; 2025), but generally do not modify the random data assignment that determines these local objectives. The motivation for revisiting this assignment is illustrated in Figure 1(a).

In datacenter distributed training, however, data can be repartitioned across nodes at each epoch. Since each node's data shard defines its empirical local objective, data partitioning directly shapes the resulting local objectives. This raises an algorithmic question: *Can data partitioning be tailored to reduce heterogeneity among local objectives under periodic synchronization?* The discrepancy between local and global gradients is commonly used to quantify first-order data heterogeneity (Yu et al., 2019a; Patel et al., 2024). Random partitioning is a natural default in standard data paral-



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

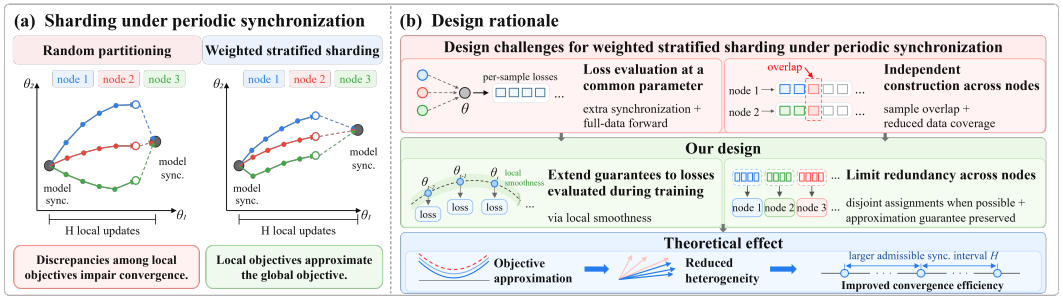



Figure 1: (a) Motivation for weighted stratified sharding under periodic synchronization. (b) Main challenges, design rationale, and theoretical implications.

lelism (Li et al., 2020), as it avoids redundant computation while drawing each node's shard from the same marginal distribution. However, when synchronization becomes infrequent, discrepancies among local objectives impair convergence. **Under random partitioning, these discrepancies can scale with the worst-case loss** (Serfling, 1974), and their effect grows with the synchronization interval (Khaled et al., 2020). Prior work has studied data partitioning through submodular objectives that model the utility of allocated data shards (Fujishige, 2005; Wei et al., 2015). Later work further considers constraints such as sample ratios and label balance (Wang et al., 2021; He & Dube, 2023). These methods optimize feature coverage through submodular utilities, but their approximation guarantees do not directly translate into bounds on the discrepancy among local objectives or the resulting convergence rate.

### 1.1 Challenges and Contributions

Intuitively, a natural way to control data heterogeneity is to make the local objective on each node approximate the global objective defined on the full dataset. This connects data sharding to importance sampling (Johnson & Guestrin, 2018) and the sensitivity-based coreset literature (Langberg & Schulman, 2010; Mirzasoleiman et al., 2020), where the sensitivity of each data point is used to construct weighted subsets with approximation guarantees. For general non-convex machine learning objectives, however, sensitivities are difficult to compute or approximate, and the required coreset size can depend on a large total sensitivity and pseudo-dimension (Huang et al., 2021; Raj et al., 2020; Padmanabhan et al., 2023). Layered sampling avoids these quantities by stratifying samples according to losses evaluated at a common parameter and reweighting samples drawn from each stratum (Huang et al., 2021). Chen et al. (2026) further tightens the error dependence to the stable rank of the training trajectory, which is often much smaller than the parameter dimension. **Nevertheless, bringing this stratification idea to the scenario of "periodically synchronized distributed training" mainly presents two obstacles**, as summarized in Figure 1(b). First, conventional stratification requires per-sample losses evaluated at a common parameter across nodes, which would require an additional model synchronization and a full-dataset forward pass at each reconstruction. Second, independently applying the single-node construction across nodes can assign the same samples to multiple nodes, causing redundant computation and reducing data coverage (Yun et al., 2022; Malinovsky et al., 2023; Demidovich et al., 2025). These challenges motivate us to propose the following goal:

*Design a weighted sharding algorithm based on stratification, where the algorithm should be able to reduce data heterogeneity and limit redundancy across nodes, while remaining efficient and compatible with distributed training pipelines.*

To this end, we exploit the local smoothness of loss functions to reuse sample losses collected during distributed training, avoiding their reevaluation at each reconstruction. Using these losses, we construct weighted stratified shards whose local objectives are unbiased estimators of the global objective and uniformly approximate it with high probability. To limit redundancy across nodes, each node samples without replacement within every stratum. Crucially, the joint approximation guarantee does not require independent sampling across nodes. This allows the algorithm to assign disjoint sample sets whenever the stratum size permits, with overlap allowed otherwise. Finally, we translate the objective approximation error into a first-order data heterogeneity bound and derive

a Local SGD convergence guarantee that links improved approximation to faster convergence and longer admissible synchronization intervals.

Overall, weighted data sharding provides a simple and effective lever for accelerating convergence under periodic synchronization. Experiments across vision and language tasks in various distributed setups show consistent improvements over random partitioning in training loss, test accuracy, and convergence speed, with negligible runtime overhead. In representative vision settings, our method achieves $1.20\times$–$1.45\times$ iteration speedups. With synchronization every five local updates, our method even outperform fully synchronized random partitioning on CIFAR-10 and ImageNet-1K while using about $80\%$ fewer synchronization rounds. In math continual pretraining on two widely used math corpora, OpenWebMath-Pro and Jiuzhang3.0, our method improves all nine downstream benchmarks. It also reaches the same perplexity $1.26\times$–$1.28\times$ faster with synchronization at only $1/500$ the frequency of fully synchronized data parallelism.

## 2 PRELIMINARIES

Let $Z = \{z_i\}_{i=1}^n$ denote the input dataset of $n$ samples, and $\theta \in \mathbb{R}^d$ be the parameter of the neural network. We consider nonnegative per-sample losses and define the objective function as $F(\theta) = \frac{1}{n}\sum_{i=1}^n f(\theta, z_i)$. We assume Lipschitz smoothness of the per-sample loss functions, which is standard in the analysis of gradient-based optimization (Wolfe, 1969).

**Assumption 2.1** (Lipschitz Smoothness)**.** There exists a real constant $L > 0$, such that for any $1 \le i \le n$ and any $\theta_1, \theta_2$ in the parameter space, we have

$$\|\nabla f(\theta_1, z_i) - \nabla f(\theta_2, z_i)\| \le L\|\theta_1 - \theta_2\|, \tag{1}$$

where $\|\cdot\|$ is the Euclidean norm in the space.

Consider a distributed training scenario with $K$ computing nodes, where each node $k \in [K] := \{1, \cdots, K\}$ is allocated a data shard $S^k$. We augment each $S^k$ with a weight vector $\mu^k = [\mu_1^k, \mu_2^k, \cdots, \mu_n^k] \in \mathbb{R}^n_{\ge 0}$. The nonzero entries of $\mu^k$ identify the samples in $S^k$, *i.e.*, $S^k = \{z_i \in Z \mid \mu_i^k > 0\}$, and each node then optimizes its *local objective function* $\tilde{F}^k(\theta) = \frac{1}{n}\sum_{z_i \in S^k} \mu_i^k f(\theta, z_i)$. Our construction chooses these weights so that each $\tilde{F}^k(\theta)$ serves as an unbiased estimator of $F(\theta)$ and satisfies a high probability bound on its approximation error. These guarantees do not depend on a specific choice of optimizer, and the details are deferred to Section 3.1.

Next, we introduce **Local SGD** (Stich, 2019) as a canonical instance of periodic synchronization, whose stochastic momentum variant is analyzed in Section 3.2. The Local SGD algorithm generates in parallel $K$ sequences of iterates $\{\theta_t^k\}_{t=0}^T$, $k \in [K]$, where $T$ is the number of parameter update steps. Given the communication interval $H$, Local SGD evolves the sequences $\{\theta_t^k\}_{t=0}^T$ in the following way:

$$\theta_{t+1}^k = \begin{cases} \theta_t^k - \eta \nabla \tilde{F}^k(\theta_t^k), & \text{if } t\%H \neq 0 \\ \frac{1}{K}\sum_{k=1}^K (\theta_t^k - \eta \nabla \tilde{F}^k(\theta_t^k)) & \text{if } t\%H = 0 \end{cases} \tag{2}$$

where $\eta$ is the learning rate. When $H = 1$, synchronization occurs at every iteration, recovering the standard fully synchronized data-parallel update. Beyond the standard synchronization of model parameters, analogous update mechanisms are also applicable to other stateful components, such as *outer gradients* Douillard et al. (2024) or *momentum buffers* Yu et al. (2019a).

## 3 WEIGHTED STRATIFIED SHARDING

In this section, we instantiate the weighted local objectives through stratified data sharding, and analyze the resulting approximation and convergence guarantees. We first construct the loss strata and derive the stratum-wise allocation rule, then relate the approximation error to data heterogeneity under Local SGD. For notational simplicity, we use $f_i(\theta)$ hereafter to denote $f(\theta, z_i)$.

For the theoretical construction, let $\theta_{\texttt{ref}}$ denote the most recent synchronized parameter preceding the current data sharding round. We first analyze the construction using losses evaluated at $\theta_{\texttt{ref}}$, and then extend the guarantee to account for losses cached during distributed training. To amortize the approximation error across strata and avoid dependence on the worst-case loss (Serfling, 1974),

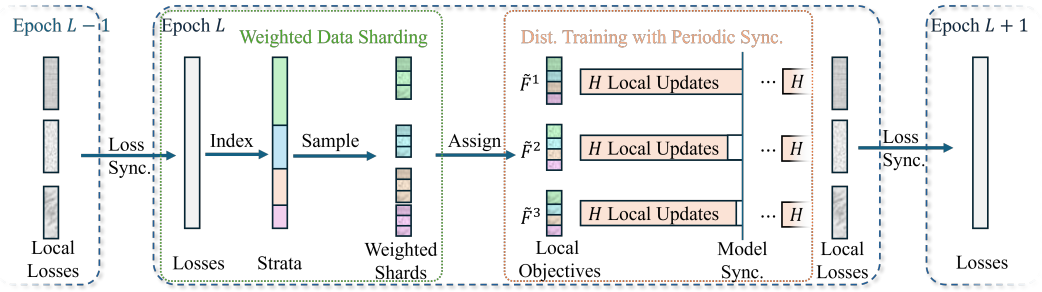



Figure 2: Overview of weighted stratified data sharding for distributed training with periodic model synchronization. Sample losses recorded during epoch $L-1$ are gathered across nodes at the start of epoch $L$ and used to form exponential strata. Weighted samples are drawn from these strata and assigned to nodes, determining each node's local objective. Model parameters are synchronized after every $H$ local updates. The resulting local objectives approximate the global objective, reducing discrepancies among nodes and improving convergence.

we group samples by exponential loss ranges relative to the average loss. Our construction uses loss stratification to control the approximation error of weighted local objectives across distributed nodes (Chen, 2009; Ding & Wang, 2020; Chen et al., 2026), without the sensitivity estimation or VC dimension dependence of importance sampling (Langberg & Schulman, 2010; Li et al., 2001; Feldman & Langberg, 2011). Formally, let $N = \lceil \log_2 n \rceil$ and $\bar{h} = F(\theta_{\texttt{ref}})$. We exponentially index the dataset $Z = \{z_i\}_{i=1}^n$ into logarithmically bounded $N+1$ strata $Z = \cup_{j=0}^N Z_j$.

$$Z_0 = \{z_i \in Z \mid f_i(\theta_{\texttt{ref}}) \le \bar{h}\}, \tag{3}$$

$$Z_j = \{z_i \in Z \mid 2^{j-1}\bar{h} < f_i(\theta_{\texttt{ref}}) \le 2^j \bar{h}\}, \quad 1 \le j \le N. \tag{4}$$

Next, for each computing node $k \in [K]$, a subset $S_j^k$ is sampled uniformly at random from the non-empty set $Z_j$ *without replacement*. The size $|S_j^k|$ is derived from the analysis in Section 3.1, so that no stratum dominates the approximation error. We then assign weight $\mu_i^k = \frac{|Z_j|}{|S_j^k|}$ to each sampled data point $z_i \in S_j^k$ and weight 0 to each unsampled point in $Z_j \setminus S_j^k$ at node $k$. The construction satisfies $\sum_{i=1}^n \mu_i^k = n$, and $\tilde{F}^k(\theta) = \frac{1}{n}\sum_{i=1}^n \mu_i^k f_i(\theta)$ is an unbiased estimator of $F(\theta)$. Figure 2 provides an overview of this weighted stratified sharding procedure and its integration into periodically synchronized distributed training.

### 3.1 Local Objective Approximation

For the approximation analysis, let $\mathbb{B}(\theta_{\texttt{ref}}, R)$ denote the Euclidean ball of radius $R$ around $\theta_{\texttt{ref}}$, and let $\mathbb{T} \subseteq \mathbb{B}(\theta_{\texttt{ref}}, R)$ be a local analysis region containing the parameter update trajectory. For a tighter characterization of the covering complexity of $\mathbb{T}$, we use the stable-rank covering bound and denote its effective dimension by $d_{\texttt{e}}$, which can be much smaller than the parameter dimension $d$ (Zhao et al., 2024; Jaiswal et al., 2025; Chen et al., 2026). $\mathcal{H}(a,b) = \frac{2ab}{a+b}$ is the harmonic mean and $\bar{h} = F(\theta_{\texttt{ref}})$. Define $M := \max\{\|\nabla f_i(\theta_{\texttt{ref}})\| \mid 1 \le i \le n\}$ and $m := \min_{\theta \in \mathbb{B}(\theta_{\texttt{ref}}, R)} F(\theta) > 0$. Let $C_{\texttt{Lip}} := MR + LR^2$ denote the Lipschitz-related term, which characterizes the possible drift of loss values within the local region of radius $R$. Here, $R$, $d_{\texttt{e}}$, $M$, and $m$ are used only in the analysis to characterize the theoretical guarantee and need not be specified individually by the algorithm.

**Stratum-wise allocation and approximation error**. We now specify the sampling budget across loss layers under the fixed per-node budget $|S^k| = \frac{n}{K}$. The following Lemma 3.1 gives the harmonic mean allocation and a uniform approximation guarantee for each weighted local objective $\tilde{F}^k$, with the error depending on the distributed scale $K$ and the loss scale $\bar{h}$ at the current training stage.

**Lemma 3.1** (Local objective approximation)**.** *Suppose the local sample size for each computing node $k$ is set to $|S^k| = \frac{n}{K}$, and the allocation for each stratum $j$ satisfies $|S_j^k| = \frac{1}{2}\mathcal{H}(|Z_j|, |\bar{S}_j|)$ up to integer rounding, with $|\bar{S}_j| \propto (\bar{h} + 2^{1-j}C_{\texttt{Lip}})^2$. Then, with probability at least $1 - 1/(nK)$, we*

*have* $\tilde{F}^k(\theta) \in (1 \pm \epsilon)F(\theta)$ *for any* $\theta \in \mathbb{T}$*, and the error bound is given by:* [1]

$$\epsilon = \tilde{O}(\tfrac{\bar{h}+C_{\mathtt{Lip}}}{m} \cdot \sqrt[4]{\tfrac{Kd_{\mathtt{e}}}{n+d_{\mathtt{e}}}}).$$

The proof treats each loss stratum as a finite population sampled without replacement and does not assume independence of the sampling events across nodes. The main ingredients for the proof are a Serfling concentration bound (Serfling, 1974) for sampling within each stratum, a covering analysis for objective approximation over the local parameter region (Chen, 2009; Huang et al., 2021), and a union bound over computing nodes. Complete details are deferred to Section B.1.

*Remark* 3.2. If the strata in Eqs. (3, 4) are constructed from cached training losses $\{f_i(\theta_{t_i})\}_{i=1}^n$ with $\|\theta_{t_i} - \theta_{\mathtt{ref}}\| = O(R)$, the same analysis applies by replacing the loss scale $\bar{h}$ with $\frac{1}{n}\sum_{i=1}^n f_i(\theta_{t_i})$. This staleness changes the Lipschitz-related term only by a constant factor. We provide the corresponding analysis in the appendix.

**Sampling budget and approximation error.** Lemma 3.1 characterizes the dependence of the approximation error on the local sampling budget. For a fixed total number of training iterations, we use $\rho \in (0, 1]$ to denote the per-epoch sampled data ratio, while keeping the same stratum-wise allocation rule in Lemma 3.1. Scaling the local sample size $|S^k|$ to $\rho|S^k|$ increases the approximation error by a factor of $\rho^{-1/4}$. The fourth-root dependence makes the error less sensitive to moderate reductions in the sample size of each node. For fixed $K$ and $d_{\mathtt{e}}$, $\epsilon$ vanishes as $n \to \infty$.

**Comparison with random partitioning.** Without exponential stratification, the same Serfling-based inequality involves the global loss range. Let $h_{\max} = \max_i f_i(\theta_{\mathtt{ref}})$. The corresponding approximation error takes the form $\epsilon_{\mathtt{rnd}} = \tilde{O}(\frac{h_{\max}+C_{\mathtt{Lip}}}{m}\sqrt[4]{\frac{Kd_{\mathtt{e}}}{n+d_{\mathtt{e}}}})$. In contrast, Lemma 3.1 controls concentration within each loss stratum, leading to a dependence on $\bar{h}$ rather than $h_{\max}$. Therefore, exponential stratification keeps the approximation error from being governed by the worst-case loss scale under uneven loss distributions.

### 3.2 Data Heterogeneity and Convergence

For convergence analysis, define the consensus objective $\tilde{F}(\theta) = \frac{1}{K}\sum_{k\in[K]} \tilde{F}^k(\theta)$ and the consensus parameter $\bar{\theta}_t = \frac{1}{K}\sum_{k\in[K]} \theta_t^k$, following standard analyses of distributed consensus optimization Lian et al. (2017); Koloskova et al. (2020); Song et al. (2022).

Building upon the objective function approximation in Lemma 3.1 and the Lipschitz smoothness in Assumption 2.1, we establish a data heterogeneity bound on interior points of the local analysis region. For $r_\epsilon \coloneqq \sqrt{2m\epsilon/L}$, define the $r_\epsilon$-interior of $\mathbb{T}$ as $\mathbb{T}_{r_\epsilon} \coloneqq \{\theta \in \mathbb{T} \mid \mathbb{B}(\theta, r_\epsilon) \subseteq \mathbb{T}\}$.

**Lemma 3.3** (First-order heterogeneity)**.** *Given* $\epsilon \in (0, 1)$ *and the sample size specified by Lemma 3.1, for any node* $k \in [K]$*, with probability at least* $1 - 1/(nK)$*, the bound* $\|\nabla\tilde{F}^k(\theta) - \nabla F(\theta)\|^2 \le 8m\epsilon L$ *holds for all* $\theta \in \mathbb{T}_{r_\epsilon}$.

Lemma 3.3 translates the objective approximation error $\epsilon$ into a first-order heterogeneity bound. The detailed proof is provided in Section B.1. Furthermore, under standard assumptions in distributed stochastic optimization (Lian et al., 2015), we connect the established heterogeneity bound to the convergence of momentum Local SGD (Yu et al., 2019a) and derive a convergence bound in terms of the sharding-induced approximation error.

**Convergence setting.** Consider a local training phase of length $T \ge 1$ covered by the analysis region $\mathbb{T} \subseteq \mathbb{B}(\theta_{\mathtt{ref}}, R)$ introduced in Section 3.1. Here, a local training phase refers to the interval between two consecutive data-shard reconstructions, during which the local objectives $\{\tilde{F}^k\}_{k=1}^K$ remain fixed. Throughout this phase, the worker iterates satisfy $\theta_t^k \in \mathbb{T}$ for all $k, t$, and the consensus iterates satisfy $\bar{\theta}_t \in \mathbb{T}_{r_\epsilon}$ for all $0 \le t < T$. Given the local objective functions $\{\tilde{F}^k\}_{k=1}^K$, assume each worker independently evaluates unbiased stochastic gradient $g_t^k$ satisfying $\mathbb{E}[g_t^k] = \nabla\tilde{F}^k(\theta_t^k)$, with bounded variance $\mathbb{E}[\|g_t^k - \nabla\tilde{F}^k(\theta_t^k)\|^2] \le \sigma^2$ for all $t, k$.

[1] $\tilde{O}(g) \coloneqq O\left(g \cdot \mathtt{polylog}\left(\frac{n\bar{h}KC_{\mathtt{Lip}}}{\epsilon m}\right)\right)$.

**Theorem 3.4** (Convergence efficiency)**.** *Under the convergence setting above, consider the momentum variant of Local SGD with momentum coefficient $\beta$. If the learning rate $\eta$ and the communication interval $H$ are chosen such that $0 < \eta \leq \frac{(1-\beta)^2}{(1+\beta)L}$ and $1 \leq H \leq \frac{1-\beta}{6L\eta}$, then it satisfies with probability $1 - 1/n$ that*

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[\|\nabla\tilde{F}(\bar{\theta}_t)\|^2] \leq O(\tfrac{1}{\eta T}) + O(\tfrac{\eta}{K}\sigma^2) + O(\eta^2 H\sigma^2) + O(\eta^2 H^2 m\epsilon L).$$

The proof of Theorem 3.4 applies Lemma 3.3 to each node and then takes a union bound over the $K$ node-level events, which does not require these sampling events to be independent. We then translate these guarantees into a bound on gradient heterogeneity and derive the momentum Local SGD convergence guarantee under standard assumptions for Local SGD (Yu et al., 2019a; Khaled et al., 2020; Woodworth et al., 2020a). The complete proof is provided in Section B.1. The resulting heterogeneity penalty is $O(\eta^2 H^2 m\epsilon L)$. Under the local-region condition $R = O(\min\{\frac{\bar{h}}{M}, \sqrt{\frac{\bar{h}}{L}}\})$, substituting $\epsilon$ from Lemma 3.1 gives a heterogeneity term that scales as $O(\eta^2 H^2 K^{1/4}\bar{h})$ up to logarithmic factors.

*Remark* 3.5 (Linear speedup). Our method achieves the linear-speedup rate $O(\frac{1}{\sqrt{KT}})$, so $K$ nodes reach tolerance $\xi$ in $T = O(K^{-1}\xi^{-2})$ iterations, compared with the single-node $O(\xi^{-2})$ rate. This holds for $\eta = \Theta(\sqrt{\frac{K}{T}})$ and $H \leq O(\min\{\frac{T^{1/2}}{K^{3/2}}, \frac{T^{1/4}}{K^{7/8}\bar{h}^{1/2}}\})$. Moreover, a smaller loss scale $\bar{h}$ relaxes the admissible synchronization interval $H$. Detailed derivations are deferred to Appendix B.1.

## 4 Implementation and Overhead

We integrate the proposed construction into distributed training as a lightweight sampler, without modifying the model architecture or optimizer. The full procedure is summarized in Algorithm 1.

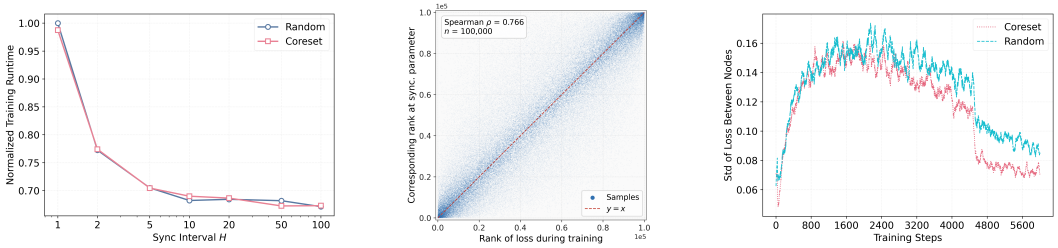
**Algorithm 1** Coreset Construction for Distributed Training

**Input:** A training dataset $Z = \{z_i\}_{i=1}^n$. Cached per-sample loss values with average $\bar{h}$. The number of computing nodes $K$. Lipschitz-related hyperparameter $C_{\texttt{Lip}}$.

1. **Setup:** Let $\mathcal{H}(a,b) = \frac{2ab}{a+b}$ denote the harmonic mean and $N = \lceil \log_2 n \rceil$. Index dataset $Z$ into $N+1$ strata $\{Z_j\}_{j=0}^N$ from the cached loss values using (3) and (4). Initialize local weight vector $\mu^k = \mathbf{0} \in \mathbb{R}^n$, and empty shard $S^k = \emptyset$.
2. **Stratified Sampling:** For each non-empty stratum $Z_j$ ($0 \leq j \leq N$), set the stratum-wise sample size $|S_j^k| \leftarrow \lceil \frac{1}{2}\mathcal{H}(|Z_j|, |\bar{S}_j|) \rceil$ according to Lemma 3.1. Here, $|\bar{S}_j| \propto (\bar{h} + 2^{1-j} \cdot C_{\texttt{Lip}})^2$, with the normalization constant set by bisection to ensure $\sum_{j=0}^N |S_j^k| = \frac{n}{K}$.
3. **Index Generation:** For each non-empty stratum $Z_j$, all nodes use a shared random seed to generate $\{I_j^k\}_{k=1}^K$, with $|I_j^k| = |S_j^k|$ as follows.
   - If $K|S_j^k| \leq |Z_j|$, draw a random permutation of indices in $Z_j$, take the first $K|S_j^k|$ indices, and split them into $K$ disjoint blocks $\{I_j^k\}_{k=1}^K$.
   - Else, draw $K$ independent index sets $\{I_j^k\}_{k=1}^K$ from the indices of $Z_j$, each uniformly without replacement.
4. **Weight Assignment:** Node $k$ prefetches $S_j^k = \{z_i \mid i \in I_j^k\}$ from the file system. Assign weight $\mu_i^k = \frac{|Z_j|}{|S_j^k|}$ for each $z_i \in S_j^k$, and set $S^k = \cup_{j=0}^N S_j^k$.

**Output:** For $k$-th node: coreset $S^k$ and the weight vector $\mu^k = [\mu_1^k, \mu_2^k, \cdots, \mu_n^k]$.

At the beginning of each epoch, nodes use cached per-sample losses to construct the loss strata, determine the stratum-wise sampling budgets, and reproduce shard assignments using a shared random seed. When the stratum size permits, index generation avoids duplicate assignments across nodes. Otherwise, nodes sample independently without replacement to satisfy the local sampling budget. The allocation granularity can be chosen according to the compute unit used as a logical




(a) Runtime comparison. (b) Rank drift under staleness. (c) Cross-node loss std.

Figure 3: Understanding Coreset for distributed training. (a) Normalized training runtime across different synchronization intervals on CIFAR-10 with ResNet-18. (b) Spearman correlation between the ranks of sample losses recorded during training and the ranks of their corresponding losses evaluated at the synchronized parameter. (c) Standard deviation of loss values across nodes during training. Panels (b) and (c) are evaluated on Tiny-ImageNet with ResNet-50 under $H = 10$.

node, such as a worker process or a multi-GPU server, which allows proportional data assignment under heterogeneous compute capacity. Implementation details are provided in Section C.

**Overhead.** Each reconstruction requires $O(n)$ time and space to process the cached losses, compute $\bar{h}$, form the strata, and generate weighted shards, while bisection contributes only logarithmic overhead (see Section B.1). Communication is limited to synchronizing the $n$-entry loss vector once per epoch. The shared seed allows nodes to reproduce shard assignments locally, avoiding centralized dispatch and its associated bandwidth bottleneck and single point of failure.

# 5 Evaluation

## 5.1 Experimental Setup

**Testbeds.** We conduct our experiments on a server equipped with 8 Nvidia RTX 6000 Ada GPUs. These GPUs are bridged via a PCIe Gen4 topology without NVLink. To compare across diverse GPU-per-node configurations, we configure a *logical node* as a set of GPUs assigned a shared hostname, supporting both symmetric and asymmetric allocations. The inter-node bandwidth through the host TCP/IP stack is 40Gbps and direct GPU peer-to-peer memory access is disabled.

**Workloads.** We include ResNet (He et al., 2016), ViT (He et al., 2022), Swin-T (Liu et al., 2021) and TinyLlama (Zhang et al., 2024) for comparisons. The datasets are CIFAR-10, CIFAR-100 (Krizhevsky & Hinton, 2009), Tiny-ImageNet (Chrabaszcz et al., 2017), ImageNet-1K (Russakovsky et al., 2015), OpenWebMath-Pro (Zhou et al., 2025) and Jiuzhang3.0 (Zhou et al., 2024).

**Baselines.** We use random partitioning as the main baseline for distributed Local SGD and denote it as "Random" in figures and tables. We report our method as "Coreset", referring to the weighted data shards constructed by Algorithm 1. Both methods regenerate their data assignments before each training epoch. We further compare with the closest prior data partitioning methods, Greedy-Facility and Greedy-GraphCut (He & Dube, 2023), in Section D. Coreset avoids their costly static construction and achieves the highest test accuracy among these partitioning baselines. The total number of training steps is aligned across compared methods.

## 5.2 Understanding Coreset for distributed training

Before presenting the main results, we first examine runtime overhead, the staleness of cached losses, and discrepancies among local objectives.

**Runtime comparisons.** Figure 3a shows that Coreset matches the runtime of Random across synchronization intervals during 200 epochs of ResNet-18 training on CIFAR-10. Increasing $H$ reduces the runtime of both methods through less frequent synchronization, and $H = 10$ lowers total runtime by up to 30% relative to Random with $H = 1$ in this setup.

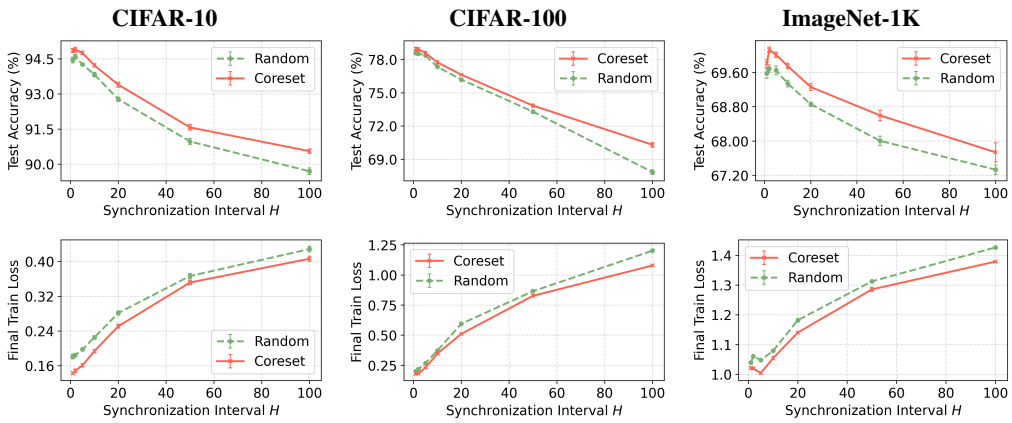



Figure 4: Test accuracy (top row) and final training loss (bottom row) under different synchronization intervals $H \in \{1, 2, 5, 10, 20, 50, 100\}$. Columns correspond to CIFAR-10 with ResNet-18, CIFAR-100 with ResNet-50, and ImageNet-1K with ResNet-50. Error bars indicate standard deviations of loss and accuracy across three random seeds.

**Bounded staleness of cached losses.** We reuse losses cached during training for exponential stratification in Eqs. (3, 4), avoiding an additional evaluation over the full dataset at each reconstruction. Remark 3.2 shows that bounded staleness can be absorbed into the local loss drift term without requiring exact agreement between the strata constructed from cached losses and those induced by losses evaluated at reconstruction. Figure 3b compares the ranks of cached losses with those of the corresponding losses evaluated at reconstruction. The concentration of points around the diagonal, together with a Spearman correlation of $0.766$, indicates limited empirical rank drift under staleness.

**Cross-node objective discrepancy.** Figure 3c reports the standard deviation of local objective losses across $K = 8$ nodes on Tiny-ImageNet, where the drop in both curves around iteration $4500$ corresponds to a learning rate decay by a factor of $0.1$. Coreset consistently has a lower cross-node standard deviation than Random, especially later in training, which is consistent with the local-objective approximation guarantee in Lemma 3.1.

### 5.3 Main Results

We present the main results for vision training under varying synchronization intervals and for LLM math continual pretraining.

**Synchronization intervals $H$.** We evaluate $H \in \{1, 2, 5, 10, 20, 50, 100\}$ on CIFAR-10, CIFAR-100, and ImageNet-1K. Figure 4 shows that Coreset consistently achieves lower training loss and higher test accuracy than Random, with an accuracy gain of over $2\%$ on CIFAR-100 at $H = 100$. At $H \in \{2, 5\}$, Coreset even outperform fully synchronized Random ($H = 1$) on CIFAR-10 and ImageNet-1K. Across the representative settings in Figure 5, Coreset achieves $1.20\times$–$1.45\times$ iteration speedups compared with Random.

**LLM math continual pretraining.** We perform full-parameter continual pretraining of TinyLlama-1.1B on mathematical data. The base model was pretrained on 3 trillion tokens of natural language and code (Zhang et al., 2024). The numbers of unique tokens and total training tokens are 5.1B and 10.2B for OpenWebMath-Pro, and 4.6B and 9.2B for Jiuzhang3.0, respectively. We use $H = 500$ for both datasets, corresponding to a $500\times$ reduction in synchronization rounds relative to per-step synchronization. Across the nine downstream math and STEM benchmarks in Table 1, Coreset improves the average score from $16.6$ to $17.9$ on OpenWebMath-Pro and from $35.1$ to $37.4$ on Jiuzhang3.0. The corresponding training curves in Figure 7 show $1.28\times$ and $1.26\times$ iteration speedups at matched perplexity on OpenWebMath-Pro and Jiuzhang3.0, respectively.

**Broader empirical validation.** Section D evaluates Coreset across node counts, compute heterogeneity, communication failures, synchronization strategies, model architectures, optimizers, label noise, and additional synchronization intervals, and compares it with greedy partitioning baselines.

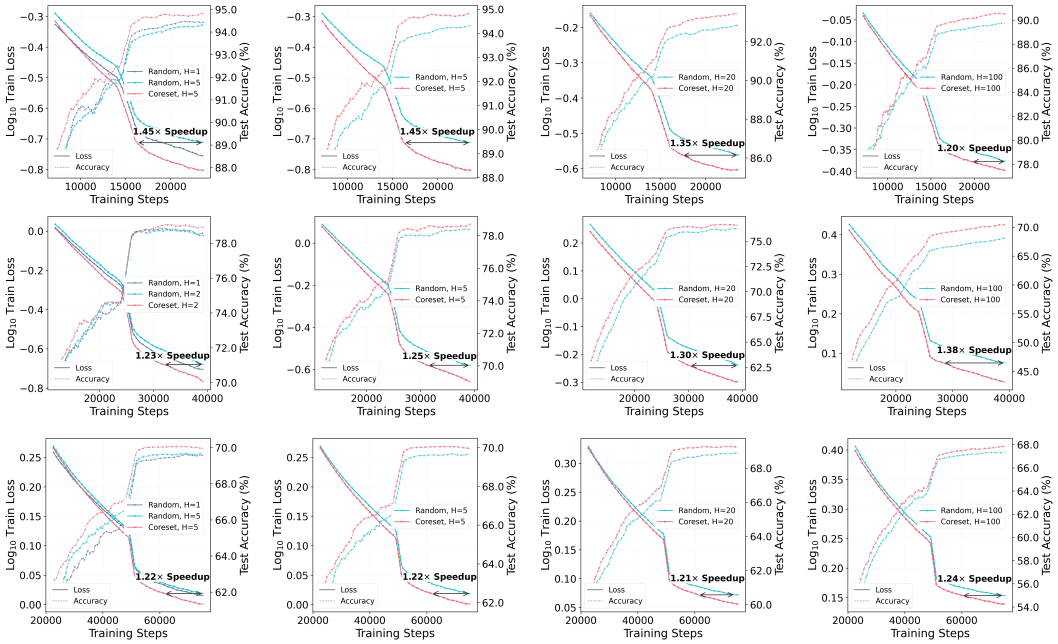



Figure 5: Convergence speedup and test accuracy across datasets and synchronization intervals. **Rows** correspond to CIFAR-10 with ResNet-18, CIFAR-100 with ResNet-50, and ImageNet-1K with ResNet-50. **Columns:** the first column shows that Coreset converges faster than Random with $H = 1$ even at larger synchronization intervals, where $H = 1$ denotes synchronization at every iteration. The remaining columns compare representative fixed intervals $H = 5$, $H = 20$, and $H = 100$. Speedups are measured in training iterations and exclude the runtime savings from less frequent synchronization.

Table 1: Evaluation of TinyLlama-1.1B before and after continual pretraining on math datasets. Avg. reports mean accuracy across nine benchmarks. Standard deviations are computed over three training runs. The best results within each dataset group are bolded.

| Dataset | Method | GSM8K | MATH | SVAMP | ASDiv | MAWPS | TAB | MQA | SAT MATH | MMLU STEM | Avg. ± std. |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Base model | | 2.7 | 2.8 | 10.9 | 17.9 | 20.5 | 12.5 | 14.0 | 21.9 | 16.3 | 13.3 |
| OpenWebMath-Pro | Random | 3.6 | 4.0 | 16.3 | 27.3 | 35.7 | 12.6 | 10.9 | 14.6 | 24.7 | 16.6 ± 0.2 |
| | Coreset | **4.1** | **4.7** | **16.9** | **28.8** | **37.2** | **13.5** | **12.0** | **18.8** | **25.5** | **17.9** ±0.2 |
| Jiuzhang3.0 | Random | 36.2 | 8.8 | 50.5 | 61.0 | 82.8 | 26.8 | 15.9 | 9.4 | 24.1 | 35.1 ± 0.3 |
| | Coreset | **36.9** | **12.2** | **52.2** | **62.2** | **83.5** | **28.0** | **16.7** | **18.8** | **26.4** | **37.4** ±0.2 |

Across these settings, Coreset consistently accelerates convergence while maintaining or improving test performance over Random, with substantially lower construction costs than the greedy baselines. Sensitivity analyses show robustness to hyperparameter choices, while ablations confirm the benefits of weighted stratification and quasi non-overlap assignment, and experiments with alternative synchronization strategies demonstrate compatibility beyond simple averaging.

## 6 Conclusion

Our analysis shows how local objective approximation induced by weighted data sharding translates into reduced first-order heterogeneity and improved convergence under periodic synchronization. Across vision and language workloads, the proposed construction consistently improves optimization and test performance with negligible runtime overhead. As distributed training continues to scale, weighted data sharding offers a promising avenue for improving training efficiency under infrequent model synchronization.

### AI USE STATEMENT

In preparing this manuscript, we used generative AI tools to assist with language editing for grammar and readability. All AI-assisted edits were reviewed by the authors. We take full responsibility for the final content of this work, including all text, claims, and artifacts produced with the assistance of generative AI.

### REPRODUCIBILITY STATEMENT

To facilitate reproducibility, we provide an anonymized source-code archive as supplementary material, together with the implementation used in our experiments. The assumptions and problem setup for the theoretical analysis are stated in the main text, and complete proofs are provided in the appendix. The Evaluation section describes the testbeds, workloads, baselines, and evaluation protocols. Further implementation details and training hyperparameters are provided in Appendix C, while hyperparameter analyses, ablations, and broader empirical results are reported in Appendix D.

## REFERENCES

Sayan Bandyapadhyay, Fedor V Fomin, and Kirill Simonov. On coresets for fair clustering in metric and euclidean spaces and their applications. *Journal of Computer and System Sciences*, 142:103506, 2024.

Zalán Borsos, Mojmir Mutny, and Andreas Krause. Coresets via bilevel optimization for continual learning and streaming. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Zachary Charles, Gabriel Teston, Lucio Dery, John Rush, Nova Fallen, Zachary Garrett, Arthur Szlam, and Arthur Douillard. Communication-efficient language model training scales reliably and robustly: Scaling laws for diloco. *Advances in Neural Information Processing Systems*, 38: 106706–106737, 2026.

Ke Chen. On coresets for k-median and k-means clustering in metric and euclidean spaces and their applications. *SIAM Journal on Computing*, 39(3):923–947, 2009.

Lei Chen, Jiawei Huang, and Hu Ding. Bridging between stable rank and data selection: A novel sampling method for fast training of deep neural networks, 2026. URL `https://openreview.net/forum?id=u6XIzTeDx3`.

Jialiang Cheng, Ning Gao, Yun Yue, Zhiling Ye, Jiadi Jiang, and Jian Sha. Edit: A local-sgd-based efficient distributed training method for large language models. In *International Conference on Learning Representations*, volume 2025, pp. 65944–65965, 2025.

Yae Jee Cho, Jianyu Wang, and Gauri Joshi. Towards understanding biased client selection in federated learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 10351–10375. PMLR, 2022.

Patryk Chrabaszcz, Ilya Loshchilov, and Frank Hutter. A downsampled variant of imagenet as an alternative to the cifar datasets. *arXiv preprint arXiv:1707.08819*, 2017.

Cody Coleman, Christopher Yeh, Stephen Mussmann, Baharan Mirzasoleiman, Peter Bailis, Percy Liang, Jure Leskovec, and Matei Zaharia. Selection via proxy: Efficient data selection for deep learning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

Laurent Condat, Artavazd Maranjyan, and Peter Richtárik. Locodl: Communication-efficient distributed learning with local training and compression. In *International Conference on Learning Representations*, volume 2025, pp. 11666–11686, 2025.

Michael Crawshaw and Mingrui Liu. Federated learning under periodic client participation and heterogeneous data: A new communication-efficient algorithm and analysis. *Advances in Neural Information Processing Systems*, 37:8240–8299, 2024.

Yury Demidovich, Petr Ostroukhov, Grigory Malinovsky, Samuel Horváth, Martin Takáč, Peter Richtárik, and Eduard Gorbunov. Methods with local steps and random reshuffling for generally smooth non-convex federated optimization. In *International Conference on Learning Representations*, volume 2025, pp. 80916–80983, 2025.

Hu Ding and Zixiu Wang. Layered sampling for robust optimization problems. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 2556–2566. PMLR, 2020.

Hadi M Dolatabadi, Sarah M Erfani, and Christopher Leckie. Adversarial coreset selection for efficient robust training. *International Journal of Computer Vision*, 131(12):3307–3331, 2023.

Haotian Dong, Jingyan Jiang, Rongwei Lu, Jiajun Luo, Jiajun Song, Bowen Li, Ying Shen, and Zhi Wang. Beyond a single ai cluster: A survey of decentralized llm training. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 29186–29200, 2025.

Arthur Douillard, Qixuan Feng, Andrei Alex Rusu, Rachita Chhaparia, Yani Donchev, Adhiguna Kuncoro, MarcAurelio Ranzato, Arthur Szlam, and Jiajun Shen. Diloco: Distributed low-communication training of language models. In *2nd Workshop on Advancing Neural Network Training: Computational Efficiency, Scalability, and Resource Optimization (WANT@ ICML 2024)*, 2024.

Arthur Douillard, Yanislav Donchev, Keith Rush, Satyen Kale, Zachary Charles, Zachary Garrett, Gabriel Teston, Dave Lacey, Ross McIlroy, Jiajun Shen, et al. Streaming diloco with overlapping communication: Towards a distributed free lunch. *arXiv preprint arXiv:2501.18512*, 2025.

Arthur Douillard, Keith Rush, Yani Donchev, Zachary Charles, Nova Fallen, Ayush Dubey, Ionel Gog, Josef Dean, Blake Woodworth, Zachary Garrett, et al. Decoupled diloco for resilient distributed pre-training. *arXiv preprint arXiv:2604.21428*, 2026.

Hubert Eichner, Tomer Koren, Brendan McMahan, Nathan Srebro, and Kunal Talwar. Semi-cyclic stochastic gradient descent. In *International Conference on Machine Learning*, pp. 1764–1773. PMLR, 2019.

Dan Feldman. Introduction to core-sets: an updated survey. *CoRR*, abs/2011.09384, 2020. URL `https://arxiv.org/abs/2011.09384`.

Dan Feldman and Michael Langberg. A unified framework for approximating and clustering data. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pp. 569–578, 2011.

Satoru Fujishige. *Submodular functions and optimization*, volume 58. Elsevier, 2005.

Margalit R Glasgow, Honglin Yuan, and Tengyu Ma. Sharp bounds for federated averaging (local sgd) and continuous perspective. In *International Conference on Artificial Intelligence and Statistics*, pp. 9050–9090. PMLR, 2022.

Xinran Gu, Kaifeng Lyu, Longbo Huang, and Sanjeev Arora. Why (and when) does local sgd generalize better than sgd? In *The Eleventh International Conference on Learning Representations*, 2023.

Farzin Haddadpour, Mohammad Mahdi Kamani, Mehrdad Mahdavi, and Viveck Cadambe. Local sgd with periodic averaging: Tighter analysis and adaptive synchronization. *Advances in Neural Information Processing Systems*, 32, 2019.

Haoze He and Parijat Dube. Rcd-sgd: Resource-constrained distributed sgd in heterogeneous environment via submodular partitioning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1393–1401, 2023.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 16000–16009, 2022.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, pp. 30016–30030, 2022.

Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv:1909.06335*, 2019.

Jiawei Huang, Ruomin Huang, Wenjie Liu, Nikolaos Freris, and Hu Ding. A novel sequential coreset method for gradient descent algorithms. In *International Conference on Machine Learning*, pp. 4412–4422. PMLR, 2021.

Lingxiao Huang, Zhize Li, Jialin Sun, and Haoyu Zhao. Coresets for vertical federated learning: Regularized linear regression and $k$-means clustering. *Advances in Neural Information Processing Systems*, 35:29566–29581, 2022a.

Ruomin Huang, Jiawei Huang, Wenjie Liu, and Hu Ding. Coresets for wasserstein distributionally robust optimization problems. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022b.

Jonathan H. Huggins, Trevor Campbell, and Tamara Broderick. Coresets for scalable bayesian logistic regression. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pp. 4080–4088, 2016.

Sami Jaghouar and Johannes Hagemann. Opendiloco: An open-source framework for globally distributed low-communication training. In *Workshop on Efficient Systems for Foundation Models II@ ICML2024*, 2024.

Sami Jaghouar, Jack Min Ong, Manveer Basra, Fares Obeid, Jannik Straube, Michael Keiblinger, Elie Bakouch, Lucas Atkins, Maziyar Panahi, Charles Goddard, et al. Intellect-1 technical report. *arXiv preprint arXiv:2412.01152*, 2024.

Ajay Jaiswal, Yifan Wang, Lu Yin, Shiwei Liu, Runjin Chen, Jiawei Zhao, Ananth Grama, Yuandong Tian, and Zhangyang Wang. From low rank gradient subspace stabilization to low-rank weights: Observations, theories, and applications. In *Forty-second International Conference on Machine Learning*, 2025.

Ziheng Jiang, Haibin Lin, Yinmin Zhong, Qi Huang, Yangrui Chen, Zhi Zhang, Yanghua Peng, Xiang Li, Cong Xie, Shibiao Nong, et al. {MegaScale}: Scaling large language model training to more than 10,000 {GPUs}. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*, pp. 745–760, 2024.

Tyler B Johnson and Carlos Guestrin. Training deep models faster with robust, approximate importance sampling. *Advances in Neural Information Processing Systems*, 31, 2018.

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.

Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International conference on machine learning*, pp. 5132–5143. PMLR, 2020.

Ahmed Khaled, Konstantin Mishchenko, and Peter Richtárik. Tighter theory for local sgd on identical and heterogeneous data. In *International conference on artificial intelligence and statistics*, pp. 4519–4529. PMLR, 2020.

Ahmed Khaled, Satyen Kale, Arthur Douillard, Chi Jin, Rob Fergus, and Manzil Zaheer. Understanding outer optimizers in local sgd: Learning rates, momentum, and acceleration. *Advances in Neural Information Processing Systems*, 38:100324–100377, 2026.

Geon-Woo Kim, Junbo Li, Shashidhar Gandham, Omar Baldonado, Adithya Gangidi, Pavan Balaji, Zhangyang Wang, and Aditya Akella. Halos: Hierarchical asynchronous local sgd over slow networks for geo-distributed large language model training. In *Forty-second International Conference on Machine Learning*, 2025.

Seong Tae Kim, Farrukh Mushtaq, and Nassir Navab. Confident coreset for active learning in medical image analysis. *CoRR*, abs/2004.02200, 2020.

Anastasia Koloskova, Nicolas Loizou, Sadra Boreiri, Martin Jaggi, and Sebastian Stich. A unified theory of decentralized sgd with changing topology and local updates. In *International conference on machine learning*, pp. 5381–5393. PMLR, 2020.

Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical Report 0, University of Toronto, Toronto, Ontario, 2009. URL https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

Michael Langberg and Leonard J. Schulman. Universal epsilon-approximators for integrals. In Moses Charikar (ed.), *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pp. 598–607. SIAM, 2010. doi: 10.1137/1.9781611973075.50. URL https://doi.org/10.1137/1.9781611973075.50.

Shen Li, Yanli Zhao, Rohan Varma, Omkar Salpekar, Pieter Noordhuis, Teng Li, Adam Paszke, Jeff Smith, Brian Vaughan, Pritam Damania, et al. Pytorch distributed: experiences on accelerating data parallel training. *Proceedings of the VLDB Endowment*, 13(12):3005–3018, 2020.

Yi Li, Philip M Long, and Aravind Srinivasan. Improved bounds on the sample complexity of learning. *Journal of Computer and System Sciences*, 62(3):516–527, 2001.

Xiangru Lian, Yijun Huang, Yuncheng Li, and Ji Liu. Asynchronous parallel stochastic gradient for nonconvex optimization. *Advances in neural information processing systems*, 28, 2015.

Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. *Advances in neural information processing systems*, 30, 2017.

Tao Lin, Sebastian U Stich, Kumar Kshitij Patel, and Martin Jaggi. Don't use large mini-batches, use local sgd. In *International Conference on Learning Representations*, 2020.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 10012–10022, 2021.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

Grigory Malinovsky, Konstantin Mishchenko, and Peter Richtárik. Server-side stepsizes and sampling without replacement provably help in federated optimization. In *Proceedings of the 4th International Workshop on Distributed Machine Learning*, pp. 85–104, 2023.

Matias Mendieta, Taojiannan Yang, Pu Wang, Minwoo Lee, Zhengming Ding, and Chen Chen. Local learning matters: Rethinking data heterogeneity in federated learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8397–8406, 2022.

Baharan Mirzasoleiman, Jeff Bilmes, and Jure Leskovec. Coresets for data-efficient training of machine learning models. In *International Conference on Machine Learning*, pp. 6950–6960. PMLR, 2020.

Tomoya Murata and Taiji Suzuki. Bias-variance reduced local sgd for less heterogeneous federated learning. In *International Conference on Machine Learning*, pp. 7872–7881. PMLR, 2021.

Hiroki Naganuma, Xinzhi Zhang, Man-Chung Yue, Ioannis Mitliagkas, Russell J Hewett, Philipp Andre Witte, and Yin Tat Lee. Pseudo-asynchronous local sgd: Robust and efficient data-parallel training. *Transactions on Machine Learning Research*, 2025.

Sasho Nedelkoski, Alexander Acker, Odej Kao, Soeren Becker, and Dominik Scheinert. Distributed low-communication training with decoupled momentum optimization. *arXiv preprint arXiv:2510.03371*, 2025.

Tristan Needham. A visual explanation of jensen's inequality. *The American mathematical monthly*, 100(8):768–771, 1993.

Jose Javier Gonzalez Ortiz, Jonathan Frankle, Mike Rabbat, Ari Morcos, and Nicolas Ballas. Tradeoffs of local sgd at scale: An empirical study. *arXiv preprint arXiv:2110.08133*, 2021.

Swati Padmanabhan, David Woodruff, and Richard Zhang. Computing approximate $\ell_p$ sensitivities. *Advances in Neural Information Processing Systems*, 36:36795–36825, 2023.

Kumar Kshitij Patel, Margalit Glasgow, Ali Zindari, Lingxiao Wang, Sebastian U Stich, Ziheng Cheng, Nirmit Joshi, and Nathan Srebro. The limits and potentials of local sgd for distributed heterogeneous learning with intermittent communication. In *The Thirty Seventh Annual Conference on Learning Theory*, pp. 4115–4157. PMLR, 2024.

Kumar Kshitij Patel, Ali Zindari, Sebastian Stich, and Lingxiao Wang. Revisiting consensus error: A fine-grained analysis of local sgd under second-order data heterogeneity. *Advances in Neural Information Processing Systems*, 38:58419–58504, 2025.

Omead Pooladzandi, David Davini, and Baharan Mirzasoleiman. Adaptive second order coresets for data-efficient machine learning. In *International Conference on Machine Learning*, pp. 17848–17869. PMLR, 2022.

Anant Raj, Cameron Musco, and Lester Mackey. Importance sampling via local sensitivity. In *International Conference on Artificial Intelligence and Statistics*, pp. 3099–3109. PMLR, 2020.

Sashank J Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečnỳ, Sanjiv Kumar, and Hugh Brendan McMahan. Adaptive federated optimization. In *International Conference on Learning Representations*, 2021.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

Lorenzo Sani, Alex Iacob, Zeyu Cao, Bill Marino, Yan Gao, Tomas Paulik, Wanru Zhao, William F Shen, Preslav Aleksandrov, Xinchi Qiu, et al. The future of large language model pre-training is federated. In *International Workshop on Federated Foundation Models in Conjunction with NeurIPS 2024*, 2024.

Lorenzo Sani, Alex Iacob, Zeyu Cao, Royson Lee, Bill Marino, Yan Gao, Wanru Zhao, Dongqi Cai, Zexi Li, Xinchi Qiu, et al. Photon: Federated llm pre-training. *Proceedings of Machine Learning and Systems*, 7, 2025.

Robert J Serfling. Probability inequalities for the sum in sampling without replacement. *The Annals of Statistics*, pp. 39–48, 1974.

Noam Shazeer, Youlong Cheng, Niki Parmar, Dustin Tran, Ashish Vaswani, Penporn Koanantakool, Peter Hawkins, HyoukJoong Lee, Mingsheng Hong, Cliff Young, et al. Mesh-tensorflow: deep learning for supercomputers. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 10435–10444, 2018.

Zhuoqing Song, Weijian Li, Kexin Jin, Lei Shi, Ming Yan, Wotao Yin, and Kun Yuan. Communication-efficient topologies for decentralized learning with $o(1)$ consensus rate. *Advances in Neural Information Processing Systems*, 35:1073–1085, 2022.

Sebastian U Stich. Local sgd converges fast and communicates little. In *International Conference on Learning Representations*, 2019.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pp. 1139–1147. pmlr, 2013.

Benjamin Thérien, Xiaolong Huang, Irina Rish, and Eugene Belilovsky. Muloco: Muon is a practical inner optimizer for diloco. In *ES-FoMo III: 3rd Workshop on Efficient Systems for Foundation Models*, 2025.

Murad Tukan, Alaa Maalouf, and Dan Feldman. Coresets for near-convex functions. *Advances in Neural Information Processing Systems*, 33:997–1009, 2020.

Jianyu Wang and Gauri Joshi. Cooperative sgd: A unified framework for the design and analysis of local-update sgd algorithms. *Journal of Machine Learning Research*, 22(213):1–50, 2021.

Shengjie Wang, Tianyi Zhou, Chandrashekhar Lavania, and Jeff A Bilmes. Constrained robust submodular partitioning. *Advances in Neural Information Processing Systems*, 34:2721–2732, 2021.

Weiyang Wang, Moein Khazraee, Zhizhen Zhong, Manya Ghobadi, Zhihao Jia, Dheevatsa Mudigere, Ying Zhang, and Anthony Kewitsch. {TopoOpt}: Co-optimizing network topology and parallelization strategy for distributed training jobs. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, pp. 739–767, 2023.

Zhuoyi Wang, Dingcheng Li, and Ping Li. Latent coreset sampling based data-free continual learning. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pp. 2077–2087, 2022.

Kai Wei, Rishabh Iyer, Shengjie Wang, Wenruo Bai, and Jeff Bilmes. How to intelligently distribute training data to multiple compute nodes: Distributed machine learning via submodular partitioning. In *Neural Information Processing Society (NIPS) Workshop, Montreal, Canada*, 2015.

Kang Wei, Jun Li, Ming Ding, Chuan Ma, Howard H Yang, Farhad Farokhi, Shi Jin, Tony QS Quek, and H Vincent Poor. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE transactions on information forensics and security*, 15:3454–3469, 2020.

Philip Wolfe. Convergence conditions for ascent methods. *SIAM review*, 11(2):226–235, 1969.

Blake Woodworth, Kumar Kshitij Patel, Sebastian Stich, Zhen Dai, Brian Bullins, Brendan Mcmahan, Ohad Shamir, and Nathan Srebro. Is local sgd better than minibatch sgd? In *International Conference on Machine Learning*, pp. 10334–10343. PMLR, 2020a.

Blake E Woodworth, Kumar Kshitij Patel, and Nati Srebro. Minibatch vs local sgd for heterogeneous distributed learning. *Advances in Neural Information Processing Systems*, 33:6281–6292, 2020b.

Haoran Xu, Jiaze Li, Wanyi Wu, and Hao Ren. Federated learning with sample-level client drift mitigation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 21752–21760, 2025.

Hao Yu, Rong Jin, and Sen Yang. On the linear speedup analysis of communication efficient momentum sgd for distributed non-convex optimization. In *International Conference on Machine Learning*, pp. 7184–7193. PMLR, 2019a.

Hao Yu, Sen Yang, and Shenghuo Zhu. Parallel restarted sgd with faster convergence and less communication: Demystifying why model averaging works for deep learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pp. 5693–5700, 2019b.

Chulhee Yun, Shashank Rajput, and Suvrit Sra. Minibatch vs local sgd with shuffling: Tight convergence bounds and beyond. In *International Conference on Learning Representations*, 2022.

Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. Tinyllama: An open-source small language model. *arXiv preprint arXiv:2401.02385*, 2024.

Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. Galore: Memory-efficient llm training by gradient low-rank projection. In *Forty-first International Conference on Machine Learning*, 2024.

Fan Zhou, Zengzhi Wang, Qian Liu, Junlong Li, and Pengfei Liu. Programming every example: Lifting pre-training data quality like experts at scale. In *International Conference on Machine Learning*, pp. 79281–79327. PMLR, 2025.

Kun Zhou, Beichen Zhang, Jiapeng Wang, Zhipeng Chen, Wayne X Zhao, Jing Sha, Zhichao Sheng, Shijin Wang, and Ji-Rong Wen. Jiuzhang3. 0: Efficiently improving mathematical reasoning by training small data synthesis models. *Advances in Neural Information Processing Systems*, 37: 1854–1889, 2024.

Ying Zhu, Yang Xu, Hongli Xu, Yunming Liao, Zhiwei Yao, and Liusheng Huang. Cross-region model training with communication-computation overlapping and delay compensation. *arXiv preprint arXiv:2504.17672*, 2025.

## A Additional Related Work

**Local SGD and federated learning.** Local SGD (Stich, 2019) and federated learning (Konečnỳ et al., 2016) are two closely related lines of work on periodic synchronization in distributed training (Dong et al., 2025). Federated learning was originally developed for communication efficient training over decentralized data (Kairouz et al., 2021), with additional concerns such as privacy protection (Wei et al., 2020) and intermittent client participation (Eichner et al., 2019; Crawshaw & Liu, 2024). When adapted to datacenter distributed training, the focus shifts mainly to the mitigation of data heterogeneity across nodes (Hsu et al., 2019; Karimireddy et al., 2020). FedBSS proposes to reduce heterogeneity by using sample loss as an empirical proxy for each sample's contribution to client drift and progressively incorporating samples with larger estimated bias during federated training (Xu et al., 2025). Many Local SGD variants can be viewed through the FedOpt framework (Reddi et al., 2021; Cho et al., 2022), which combines local optimization with a periodic outer update based on aggregated local updates. Generalized Local SGD can be recovered as a special case when the outer update is gradient descent (Khaled et al., 2026). Their communication and optimization tradeoffs have been widely studied (Yu et al., 2019b; Lin et al., 2020; Ortiz et al., 2021). The convergence and generalization of Local SGD have received extensive theoretical attention under various heterogeneity assumptions (Haddadpour et al., 2019; Yu et al., 2019a; Khaled et al., 2020; Woodworth et al., 2020a;b; Murata & Suzuki, 2021; Mendieta et al., 2022; Glasgow et al., 2022; Gu et al., 2023; Khaled et al., 2026). SCAFFOLD uses control variates to correct gradient discrepancies induced by fixed local objectives and obtains a basic convergence rate without a first order heterogeneity assumption (Karimireddy et al., 2020). Our sharding limits these gradient discrepancies throughout local training, reducing the magnitude of SCAFFOLD's control variate correction. Recent analyses further identify second order heterogeneity, together with first order conditions, as a key quantity governing the communication efficiency of Local SGD (Patel

et al., 2024; 2025). Our result derives an explicit high probability bound on first order heterogeneity from objective approximation induced by data assignment. More recently, Local SGD has also been applied to communication efficient distributed training of large language models (Douillard et al., 2024; Jaghouar & Hagemann, 2024; Cheng et al., 2025; Kim et al., 2025; Sani et al., 2025; Thérien et al., 2025; Condat et al., 2025; Douillard et al., 2026; Charles et al., 2026). Our work targets a complementary design dimension: weighted data sharding directly shapes the local objectives optimized between synchronizations. We refer readers to Kairouz et al. (2021) for a broader survey.

**Data selection.** Data selection provides two relevant principles for constructing informative subsets or shards. The first is greedy selection, often formulated through submodular objectives that reflect data coverage, diversity, or representativeness. In distributed training, the works most closely related to ours are Wei et al. (2015) and He & Dube (2023), which study data partitioning across workers via submodular optimization. Submodular objectives provide a flexible modeling framework, whose guarantees target coverage or representativeness utilities rather than approximation of the training objective or the induced gradient heterogeneity. The second principle is importance sampling, which constructs weighted subsets, often called coresets, by sampling according to data point sensitivity (Feldman, 2020; Langberg & Schulman, 2010; Mirzasoleiman et al., 2020). Coresets have been studied for a broad range of optimization and learning problems, including clustering (Chen, 2009; Bandyapadhyay et al., 2024), regression (Huggins et al., 2016; Tukan et al., 2020; Huang et al., 2022a), robust optimization (Ding & Wang, 2020; Huang et al., 2022b), active learning (Coleman et al., 2020; Kim et al., 2020), robust training (Mirzasoleiman et al., 2020; Dolatabadi et al., 2023), and continual learning (Borsos et al., 2020; Wang et al., 2022). For general neural network objectives, however, sensitivity is hard to compute or approximate (Raj et al., 2020; Padmanabhan et al., 2023), and existing guarantees often scale at least linearly with the parameter dimension (Tukan et al., 2020; Mirzasoleiman et al., 2020; Huang et al., 2021; Pooladzandi et al., 2022). Moreover, these methods are mainly developed for single-node subset selection and commonly rely on sampling with replacement, which can induce substantial overlap if directly applied to distributed sharding.

# B Proofs

## B.1 Theoretical Guarantee for Coreset construction

We use the following bounds (5) to estimate the range of fluctuations in loss value of each data point lying in different loss regions.

$$f_i(\theta) \in f_i(\theta_{\texttt{ref}}) \pm \left( \frac{\|\nabla f_i(\theta_{\texttt{ref}})\|}{R} + \frac{L}{2} \right) R^2. \quad \forall \theta \in \mathbb{T} \subseteq \mathbb{B}(\theta_{\texttt{ref}}, R), \quad 1 \le i \le n \tag{5}$$

Bounds (5) could be obtained by combining Taylor expansion and Assumption 2.1.

*Claim* 1. $\sum_{j=0}^{N} |Z_j| 2^j \le 3n$.

*Proof.* By the definition of $Z_j$, we have

$$\begin{aligned} &2^j \bar{h} = \bar{h}, && \text{if } j = 0; \\ &2^j \bar{h} \le 2 f_i(\theta_{\texttt{ref}}), \forall z_i \in Z_j, && \text{if } j \ge 1. \end{aligned} \tag{6}$$

Therefore, $2^j \bar{h}$ is always no larger than $2 f_i(\theta_{\texttt{ref}}) + \bar{h}$ for any $0 \le j \le N$ and any $z_i \in Z_j$.

Overall,

$$\begin{aligned} \sum_{j=0}^{N} |Z_j| 2^j \bar{h} &= \sum_{j=0}^{N} \sum_{z_i \in Z_j} 2^j \bar{h} \\ &\le \sum_{j=0}^{N} \sum_{z_i \in Z_j} (2 f_i(\theta_{\texttt{ref}}) + \bar{h}) \\ &= 2nF(\theta_{\texttt{ref}}) + n\bar{h} = 3n\bar{h}. \end{aligned} \tag{7}$$

Thus the claim $\sum_{j=0}^{N} |Z_j| 2^j \le 3n$ is true. □

**Lemma B.1.** *Given $\epsilon_1 \in (0,1)$, $\epsilon_2 > 0$, and the number of computing nodes $K \geq 1$. For each stratum $j$, set $\delta = \epsilon_2 2^{j-1}\bar{h}$. Suppose the sample size drawn by node $k$ from stratum $Z_j$ in Algorithm 1 satisfies*

$$|S_j^k| = \left\lceil \frac{1}{2}\mathcal{H}(|Z_j|, |\bar{S}_j|) \right\rceil, \qquad |\bar{S}_j| = O\left((2^{j-1}\bar{h} + MR + LR^2)^2 \delta^{-2} \log \frac{1}{\epsilon_1}\right). \tag{8}$$

*Then for any fixed $\theta \in \mathbb{T} \subseteq \mathbb{B}(\theta_{\texttt{ref}}, R)$,*

$$\left|\tilde{F}^k(\theta) - F(\theta)\right| \leq \frac{3}{2}\epsilon_2 F(\theta_{\texttt{ref}}) \tag{9}$$

*holds with probability at least $1 - (N+1)\epsilon_1$.*

*Proof.* We first consider $1 \leq j \leq N$. The loss values $\{f_i(\theta) \mid z_i \in Z_j\}$ form a finite population, and $S_j^k$ is sampled from $Z_j$ without replacement. Through the partition construction (3) and (4), and the bounds (5), we have

$$\left.\begin{aligned} f_i(\theta) &\geq 2^{j-1}\bar{h} - MR - \frac{1}{2}LR^2; \\ f_i(\theta) &\leq 2^j\bar{h} + MR + \frac{1}{2}LR^2. \end{aligned}\right\} \tag{10}$$

Through the Serfling's inequality (Serfling, 1974), we know that

$$\texttt{Prob}\left[\left|\frac{1}{|S_j^k|}\sum_{z_i \in S_j^k} f_i(\theta) - \frac{1}{|Z_j|}\sum_{z_i \in Z_j} f_i(\theta)\right| \geq \delta\right]$$

is no larger than $2e^{-\frac{2|S_j^k|\delta^2}{(1-\frac{|S_j^k|-1}{|Z_j|})(2^{j-1}\bar{h}+LR^2+2MR)^2}}$.

Let $a = |Z_j|$, $b = |\bar{S}_j|$, and $s = |S_j^k| = \lceil ab/(a+b) \rceil$. We choose $b = \left\lceil (2^{j-1}\bar{h} + LR^2 + 2MR)^2 \delta^{-2} \log \frac{2}{\epsilon_1} \right\rceil$. If $s = a$, the stratum is sampled in full and the deviation is zero. Otherwise, $s \leq a - 1$. Since $s \geq ab/(a+b)$, the latter implies $b \leq a(a-1)$, and hence

$$\frac{s}{1 - \frac{s-1}{a}} = \frac{as}{a - s + 1} \geq \frac{a^2 b}{a^2 + a + b} \geq \frac{b}{2}. \tag{11}$$

Substituting this bound into Serfling's inequality gives a failure probability at most $2\exp\left(-b\delta^2/(2^{j-1}\bar{h} + LR^2 + 2MR)^2\right) \leq \epsilon_1$.

Now we consider the case $j = 0$. For any given data item $z_i \in Z_0$, we have $0 \leq f_i(\theta) \leq \bar{h} + MR + \frac{1}{2}LR^2$. Setting $|\bar{S}_0| = \left\lceil (\bar{h} + \frac{1}{2}LR^2 + MR)^2 \delta^{-2} \log \frac{2}{\epsilon_1} \right\rceil$ and $|S_0^k| = \left\lceil \frac{1}{2}\mathcal{H}(|Z_0|, |\bar{S}_0|) \right\rceil$, the same argument gives a failure probability at most $\epsilon_1$.

Combining the two cases above and plugging in the value of $\delta$, it holds with probability at most $\epsilon_1$ that

$$\left|\frac{|Z_j|}{|S_j^k|}\sum_{z_i \in S_j^k} f_i(\theta) - \sum_{z_i \in Z_j} f_i(\theta)\right| \geq |Z_j|\epsilon_2 2^{j-1}\bar{h}. \tag{12}$$

Recall $\tilde{F}^k(\theta) = \frac{1}{n}\sum_{i=1}^n \mu_i^k f_i(\theta)$, where for each $z_i \in Z_j$, $\mu_i^k = \frac{|Z_j|}{|S_j^k|}$ if $z_i \in S_j^k$, and $\mu_i^k = 0$ if $z_i \in Z_j \setminus S_j^k$. Thus, by taking the union bound of (12) over $0 \le j \le N$, we have

$$\begin{aligned} & n\left|\tilde{F}^k(\theta) - F(\theta)\right| \\ = & \left|\sum_{j=0}^N \frac{|Z_j|}{|S_j^k|}\sum_{z_i\in S_j^k} f_i(\theta) - \sum_{j=0}^N \sum_{z_i\in Z_j} f_i(\theta)\right| \\ \le & \sum_{j=0}^N \left|\frac{|Z_j|}{|S_j^k|}\sum_{z_i\in S_j^k} f_i(\theta) - \sum_{z_i\in Z_j} f_i(\theta)\right| \\ \le & \sum_{j=0}^N |Z_j|\epsilon_2 2^{j-1}\bar{h} \end{aligned} \tag{13}$$

with probability at least $1-(N+1)\epsilon_1$.

By using Claim 1, (13) can be rewritten as

$$n\left|\tilde{F}^k(\theta) - F(\theta)\right| \le \frac{3}{2}\epsilon_2 n F(\theta_{\texttt{ref}}). \tag{14}$$

So we complete the proof.

□

**Proof of Remark 3.2.**

*Proof.* It suffices to adapt the fixed-parameter concentration step in Lemma B.1. Let $\ell_i = f_i(\theta_{t_i})$ denote the cached loss used for stratification, and let $\hat{h} = \frac{1}{n}\sum_{i=1}^n \ell_i$. We construct the strata $\{Z_j\}_{j=0}^N$ by replacing $f_i(\theta_{\texttt{ref}})$ and $\bar{h}$ in Eqs. (3, 4) with $\ell_i$ and $\hat{h}$, respectively. Throughout this proof, $|Z_j|$ denotes the stratum size induced by this cached-loss partition. The counting bound in Claim 1 is unchanged. Specifically, for any $z_i \in Z_j$,

$$2^j\hat{h} \le 2\ell_i + \hat{h}, \qquad 0 \le j \le N, \tag{15}$$

where the inequality is trivial for $j = 0$, and follows from the stratum definition for $j \ge 1$. Therefore,

$$\sum_{j=0}^N |Z_j| 2^j \hat{h} \le \sum_{j=0}^N \sum_{z_i\in Z_j} (2\ell_i + \hat{h}) = 3n\hat{h}, \tag{16}$$

which implies that the same counting bound $\sum_{j=0}^N |Z_j|2^j \le 3n$ holds for the exponential stratification based on cached losses. This accounts for the possible change in the stratum sizes caused by using cached losses.

It remains to check the range used in Serfling's inequality. By the staleness condition, there exists a constant $c_{\texttt{st}} > 0$ such that $\|\theta_{t_i} - \theta_{\texttt{ref}}\| \le c_{\texttt{st}} R$. As a concrete case, suppose the cached losses are collected during the epoch immediately preceding the reconstruction at $\theta_{\texttt{ref}}$. Let $\theta_{\texttt{ref}}^{\texttt{old}}$ denote the synchronized parameter at the beginning of that preceding epoch. If the preceding local trajectory is contained in $\mathbb{B}(\theta_{\texttt{ref}}^{\texttt{old}}, R)$, then both $\theta_{t_i}$ and the current reconstruction parameter $\theta_{\texttt{ref}}$ lie in this ball. The triangle inequality gives $\|\theta_{t_i} - \theta_{\texttt{ref}}\| \le 2R$, corresponding to $c_{\texttt{st}} = 2$. By the same Taylor expansion used in Eq. (5), we have

$$|f_i(\theta_{t_i}) - f_i(\theta_{\texttt{ref}})| \le c_{\texttt{st}} MR + \frac{c_{\texttt{st}}^2}{2} LR^2. \tag{17}$$

Combining this bound with Eq. (5), for any $\theta \in \mathbb{T} \subseteq \mathbb{B}(\theta_{\texttt{ref}}, R)$,

$$|f_i(\theta) - \ell_i| \le (1 + c_{\texttt{st}})MR + \frac{1 + c_{\texttt{st}}^2}{2} LR^2. \tag{18}$$

Denote the right-hand side by $C^{\mathtt{st}}_{\mathtt{Lip}}$. Since $c_{\mathtt{st}}$ is a constant, $C^{\mathtt{st}}_{\mathtt{Lip}}$ is a constant-factor enlargement of $C_{\mathtt{Lip}} = MR + LR^2$. Hence, for $1 \le j \le N$ and $z_i \in Z_j$,

$$\left.\begin{aligned} f_i(\theta) &\ge 2^{j-1}\hat{h} - C^{\mathtt{st}}_{\mathtt{Lip}}; \\ f_i(\theta) &\le 2^{j}\hat{h} + C^{\mathtt{st}}_{\mathtt{Lip}}. \end{aligned}\right\} \tag{19}$$

Compared with Eq. (10), this only replaces the Lipschitz-related range term by a constant-factor enlargement, while replacing the loss scale $\bar{h}$ with $\hat{h}$. The case $j = 0$ follows in the same way with $0 \le f_i(\theta) \le \hat{h} + C^{\mathtt{st}}_{\mathtt{Lip}}$. Therefore, the Serfling argument in Lemma B.1, and the subsequent union-bound and covering arguments, apply without changing the asymptotic rate of the error bound.

□

**Lemma B.2.** *Let $V \subseteq \mathbb{T}$ be a finite set. Suppose $\epsilon_2 \ge 0$. In the sample size (8) of Lemma B.1, we set $\delta = \epsilon_2 2^{j-1}\bar{h}$ for $j = 0, 1, \cdots, N$, respectively, and replace $\epsilon_1$ in equation 8 by $\frac{\epsilon_1}{(N+1)|V|}$. The following*

$$\left|\tilde{F}^k(\theta') - F(\theta')\right| \le \frac{3}{2}\epsilon_2 F(\theta_{\mathtt{ref}}) \tag{20}$$

*holds for all $\theta' \in V$ with probability at least $1 - \epsilon_1$.*

*Proof.* Lemma B.1 shows that, for a single $\theta$, $|\tilde{F}^k(\theta) - F(\theta)| > \frac{3}{2}\epsilon_2 F(\theta_{\mathtt{ref}})$ holds with probability at most $(N+1)\epsilon_1$. By replacing $\epsilon_1$ with $\frac{\epsilon_1}{(N+1)|V|}$ in equation 8, the failure probability for any fixed $\theta' \in V$ is at most $\frac{\epsilon_1}{|V|}$. Taking a union bound over all $\theta' \in V$, $|\tilde{F}^k(\theta') - F(\theta')| \le \frac{3}{2}\epsilon_2 F(\theta_{\mathtt{ref}})$ holds with probability at least $1 - \epsilon_1$.

□

**Lemma B.3.** *Define $M' := \max_{1\le i\le n} \max_{\theta\in\mathcal{B}(\theta_{\mathtt{ref}},R)} \|\nabla f_i(\theta)\|$. Set $\epsilon_2 = \frac{2m\epsilon}{7F(\theta_{\mathtt{ref}})}$ in Lemma B.2, where $m = \min_{\theta\in\mathbb{B}(\theta_{\mathtt{ref}},R)} F(\theta)$. Set $\epsilon_3 = \frac{2\epsilon_2 F(\theta_{\mathtt{ref}})}{R\left(\sqrt{M'^2 + 2L\epsilon_2 F(\theta_{\mathtt{ref}})} + M'\right)}$. Let $V \subseteq \mathbb{T}$ be a finite anchor set such that for every $\theta \in \mathbb{T}$, there exists $\theta' \in V$ with $\|\theta - \theta'\| \le \epsilon_3 R$. The following*

$$|\tilde{F}^k(\theta) - F(\theta)| \le \epsilon F(\theta) \tag{21}$$

*holds for all $\theta \in \mathbb{T} \subseteq \mathbb{B}(\theta_{\mathtt{ref}}, R)$, with probability at least $1 - \epsilon_1$.*

*Proof.* By the stable-rank covering bound (Chen et al., 2026), such an anchor set can be chosen with

$$|V| = O\left(\left(\frac{2\sqrt{\pi e}}{\epsilon_3}\right)^{d_e/\epsilon_3^2}\right). \tag{22}$$

By the definition of the anchor set, for any $\theta \in \mathbb{T}$, there exists $\theta' \in V$ such that $\|\theta - \theta'\| \le \epsilon_3 R$. By using the similar manner of equation 5, for any $1 \le i \le n$, we have

$$\left|f_i(\theta) - f_i(\theta')\right| \le \epsilon_3 M' R + \frac{1}{2}L\epsilon_3^2 R^2. \tag{23}$$

This implies both

$$|F(\theta) - F(\theta')| \text{ and } |\tilde{F}^k(\theta) - \tilde{F}^k(\theta')| \quad \le \quad \epsilon_3 M' R + \frac{1}{2}L\epsilon_3^2 R^2. \tag{24}$$

Combining Lemma B.2, bounds (24) and triangle inequality, we obtain

$$\begin{aligned} |\tilde{F}^k(\theta) - F(\theta)| &\le |\tilde{F}^k(\theta) - \tilde{F}^k(\theta')| + |\tilde{F}^k(\theta') - F(\theta')| + |F(\theta') - F(\theta)| \\ &\le \frac{3}{2}\epsilon_2 F(\theta_{\mathtt{ref}}) + 2 \times (\epsilon_3 M' R + \frac{1}{2}L\epsilon_3^2 R^2), \end{aligned} \tag{25}$$

By letting $\epsilon_2 = \frac{2m\epsilon}{7F(\theta_{\mathtt{ref}})}$ and $\epsilon_3 = \frac{2\epsilon_2 F(\theta_{\mathtt{ref}})}{R\left(\sqrt{M'^2 + 2L\epsilon_2 F(\theta_{\mathtt{ref}})} + M'\right)}$, we have $|\tilde{F}^k(\theta) - F(\theta)| \le m\epsilon \le \epsilon F(\theta)$ via simple calculations. □

**Proof of Lemma 3.1.**

*Proof.* We use Lemma B.1 to provide loss approximation for a given $\theta$. Then we use Lemma B.2 to generalize the result to the set containing all anchors. Finally, by using the local Lipschitz property, we show that loss approximation guarantee is valid for every $\theta \in \mathbb{T}$ by Lemma B.3.

Last, we specify the obtained weighted subset size, *i.e.,* the number of non-zero entries of $\mu^k = [\mu_1^k, \mu_2^k, \cdots, \mu_n^k]$ returned by Algorithm 1. To guarantee the success probability to be at least $1 - 1/(nK)$, we set $\epsilon_1 = 1/(nK)$. Then for each computing node $k$, we first denote the intermediate quantity for each stratum $j$ as

$$|\bar{S}_j| \propto (\bar{h} + 2^{1-j}(MR + LR^2))^2, \quad 0 \le j \le N. \tag{26}$$

By the allocation in Algorithm 1,

$$|S_j^k| = \left\lceil \frac{1}{2}\mathcal{H}(|Z_j|, |\bar{S}_j|) \right\rceil \lesssim \frac{|\bar{S}_j|}{1 + |\bar{S}_j|/|Z_j|}, \tag{27}$$

where integer rounding changes the sample size by at most a constant factor for non-empty strata. And the overall sample size for computing node $k$, *i.e.,* the number of non-zero entries of $\mu^k$, equals $\sum_{j=0}^N |S_j^k|$. By applying Jensen's inequality (Needham, 1993), we establish the upper bound on the summation:

$$\sum_{j=0}^N |S_j^k| \lesssim \sum_{j=0}^N \frac{|\bar{S}_j|}{1 + |\bar{S}_j|/|Z_j|} \le \frac{\sum_{j=0}^N |\bar{S}_j|}{1 + \frac{\sum_{j=0}^N |\bar{S}_j|}{\sum_{j=0}^N |Z_j|}} = \mathtt{sub}_n\Big(\sum_{j=0}^N |\bar{S}_j|\Big), \tag{28}$$

where $n = \sum_{j=0}^N |Z_j|$ is the dataset size, $\mathtt{sub}_n(x) \coloneqq \frac{x}{1+x/n}$ grows sublinearly with its input $x$ and saturates at $n$. We use $\mathtt{sub}_n(c \cdot x) \le c \cdot \mathtt{sub}_n(x)$ for any $c \ge 1$, together with $M'R \le MR + LR^2 \le \bar{h} + MR + LR^2$ from Assumption 2.1. Combining equation 8, the setting of $\delta$ in Lemma B.1, the definitions of $\epsilon_2, \epsilon_3$ in Lemma B.3, the covering bound equation 22, and the choice of $\epsilon_1$ in Lemma B.2, we obtain

$$\sum_{j=0}^N |S_j^k| = \tilde{O}\left(\left(\frac{\bar{h} + MR + LR^2}{m}\right)^4 \cdot \frac{\mathtt{sub}_n(d_\mathtt{e})}{\epsilon^4}\right). \tag{29}$$

By substituting $|S^k| = \sum_{j=0}^N |S_j^k| = \frac{n}{K}$, $\mathtt{sub}_n(x) = \frac{x}{1+x/n}$ into equation 29, and rearranging the inequality to solve for the approximation error $\epsilon$, we bound the deviation of local objective function given a fixed per-node sampling budget:

$$\epsilon = \tilde{O}\left(\frac{\bar{h} + MR + LR^2}{m} \cdot \sqrt[4]{\frac{K d_\mathtt{e}}{n + d_\mathtt{e}}}\right). \tag{30}$$

□

**Proof of Lemma 3.3.**

*Proof.* Let $h^k(\theta) \coloneqq F(\theta) - \tilde{F}^k(\theta)$ denote the local objective function gap at node $k \in [K]$. By Lemma 3.1 and Lemma B.3, for each data shard $S^k$, the condition $|h^k(\theta)| \le m\epsilon$ holds for any $\theta \in \mathbb{T}$ with probability $1 - 1/(nK)$.

Under Assumption 2.1, each $f_i$ is $L$-smooth. Since $\mu_i^k \ge 0$ and $\sum_{i=1}^n \mu_i^k = n$, both $F(\theta) = \frac{1}{n}\sum_{i=1}^n f_i(\theta)$ and $\tilde{F}^k(\theta) = \frac{1}{n}\sum_{i=1}^n \mu_i^k f_i(\theta)$ are $L$-smooth. Consequently, $h^k(\theta)$ is $2L$-smooth. Therefore, for any $\theta \in \mathbb{T}_{r_\epsilon}$ and any perturbation $u$ satisfying $\|u\| \le r_\epsilon$, the smoothness inequality gives

$$\left|h^k(\theta + u) - h^k(\theta) - \nabla h^k(\theta)^\top u\right| \le L\|u\|^2. \tag{31}$$

Rearranging the terms and using $|h^k(\theta)| \le m\epsilon$ for all $\theta \in \mathbb{T}$, we obtain

$$\begin{aligned} |\nabla h^k(\theta)^\top u| &\le |h^k(\theta + u)| + |h^k(\theta)| + L\|u\|^2 \\ &\le 2m\epsilon + L\|u\|^2, \end{aligned} \tag{32}$$

where $\theta + u \in \mathbb{T}$ by the definition of $\mathbb{T}_{r_\epsilon}$. If $\nabla h^k(\theta) = 0$, the desired bound is trivial. Otherwise, set $u = r_\epsilon \nabla h^k(\theta)/\|\nabla h^k(\theta)\|$. Then $\|u\| = r_\epsilon$ and $\theta + u \in \mathbb{T}$, so equation 32 gives

$$r_\epsilon \|\nabla h^k(\theta)\| \leq 2m\epsilon + Lr_\epsilon^2 = 4m\epsilon. \tag{33}$$

Since $r_\epsilon = \sqrt{2m\epsilon/L}$, we obtain

$$\|\nabla h^k(\theta)\| \leq 2\sqrt{2m\epsilon L}. \tag{34}$$

That is:

$$\|\nabla \tilde{F}^k(\theta) - \nabla F(\theta)\|^2 \leq 8m\epsilon L. \tag{35}$$

□

**Proof of Theorem 3.4.**

*Proof.* By Lemma 3.3, for each node $k \in [K]$, the condition $\|\nabla \tilde{F}^k(\bar{\theta}_t) - \nabla F(\bar{\theta}_t)\|^2 \leq 8m\epsilon L$ holds with probability $1 - 1/(nK)$. Utilizing the union bound, this condition holds simultaneously across all $K$ nodes with probability at least $1 - 1/n$, which directly implies

$$\frac{1}{K}\sum_{k=1}^{K} \|\nabla \tilde{F}^k(\bar{\theta}_t) - \nabla F(\bar{\theta}_t)\|^2 \leq 8m\epsilon L. \tag{36}$$

Since $\nabla \tilde{F}(\bar{\theta}_t) = \frac{1}{K}\sum_{k=1}^{K} \nabla \tilde{F}^k(\bar{\theta}_t)$, the sum of squared distances to the mean $\nabla \tilde{F}(\bar{\theta}_t)$ is no larger than the sum of squared distances to any reference vector, such as $\nabla F(\bar{\theta}_t)$. Therefore, we obtain

$$\frac{1}{K}\sum_{k=1}^{K} \|\nabla \tilde{F}^k(\bar{\theta}_t) - \nabla \tilde{F}(\bar{\theta}_t)\|^2 \leq \frac{1}{K}\sum_{k=1}^{K} \|\nabla \tilde{F}^k(\bar{\theta}_t) - \nabla F(\bar{\theta}_t)\|^2 \leq 8m\epsilon L. \tag{37}$$

Equation 37 bounds the cross-node gradient heterogeneity by $8m\epsilon L$. Under the local-region condition $\theta_t^k \in \mathbb{T}$ and $\bar{\theta}_t \in \mathbb{T}_{r_\epsilon}$ stated in Theorem 3.4, Theorem 1 of Yu et al. (2019a) therefore applies to the weighted local objectives with $f_i$, $f$, $x_i^{(t)}$, $\bar{x}^{(t)}$, and $\kappa^2$ corresponding to $\tilde{F}^k$, $\tilde{F}$, $\theta_t^k$, $\bar{\theta}_t$, and $8m\epsilon L$, respectively, yielding Theorem 3.4. □

**Proof of Remark 3.5.**

*Proof.* **Linear speedup** under an appropriate synchronization interval. From Theorem 3.4, choosing $\eta = \Theta(\sqrt{\frac{K}{T}})$ within the stepsize range balances the first two terms at $O(\frac{1}{\sqrt{KT}})$, while requiring the remaining two terms to be no larger than this rate gives $H \leq O(\min\{\frac{T^{1/2}}{K^{3/2}}, \frac{T^{1/4}}{K^{7/8}\bar{h}^{1/2}}\})$. Under these choices, we have $\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}[\|\nabla \tilde{F}(\bar{\theta}_t)\|^2] = O(\frac{1}{\sqrt{KT}})$. Thus, $K$ nodes reach tolerance $\xi$ in $T = O(K^{-1}\xi^{-2})$ iterations, establishing the linear speedup in $K$ over the single-node $O(\xi^{-2})$ rate. Moreover, when $\bar{h} \gtrsim K^{5/4}/\sqrt{T}$, the loss-dependent term determines the admissible scale of $H$. A smaller loss scale relaxes the upper bound on $H$ under the proposed Coreset construction, with $T$ local iterations requiring $T/H$ synchronization rounds in total. □

**Computational complexity.**

*Proof.* The runtime of Algorithm 1 is $O(n)$. The dominant term comes from a constant number of passes over the per-sample loss values, which are by-products of training. These passes compute the average loss $\bar{h}$, partition the dataset according to (3) and (4), and assign weights to selected samples. The bisection determines the normalization constant of $|\bar{S}_j|$ so that the budget constraint $\sum_{j=0}^{N} |S_j^k| = n/K$ is satisfied. Let $I_{\texttt{bis}}$ denote the number of bisection iterations. Since the search operates only over the $N + 1 = O(\log n)$ loss strata, each bisection step evaluates $\sum_{j=0}^{N} |S_j^k|$ in $O(N) = O(\log n)$ time. If the searched scale lies in $[\phi_{\min}, \phi_{\max}]$ and the target interval width is $\tau$, then $I_{\texttt{bis}} = \lceil \log_2((\phi_{\max} - \phi_{\min})/\tau) \rceil$. In typical fixed-precision implementations, $I_{\texttt{bis}}$ is a

small constant: 32 bisection steps shrink the search interval by $2^{-32}$, which is sufficient for single-precision allocation, while 64 steps shrink it by $2^{-64}$ for double precision. Hence the allocation overhead is $O(\log n \cdot I_{\texttt{bis}}) = O(\log n)$ for fixed precision and is dominated by the $O(n)$ data pass.

□

## C Implementation Details

**Periodic synchronization.** To implement Local SGD, we define the intra-node and inter-node communication groups via `torch.distributed.new_group`, and we customize communication hooks via `register_comm_hook`. This allows us to modify the default gradient synchronization of `DistributedDataParallel`, and periodically `All-Reduce` parameter updates across nodes.

**Sample loss values communication.** Stratified sampling relies on per-sample loss values. We record these losses during training forward passes and share the resulting loss tensor across nodes using the *All-Gather* collective. The sampling weights, provided by the `DataLoader`, are then used to compute the weighted local objective function before invoking the PyTorch C++ autograd engine. At initialization, we set all loss values to zero, and all samples belong to a single stratum for the first epoch.

**Budget allocation via bisection.** The harmonic mean allocation adopted in Algorithm 1 features a soft-minimum behavior, ensuring $|S_j^k| \leq |Z_j|$ for every non-empty stratum. In implementation, we determine the normalization constant of $|\bar{S}_j|$ by solving the local budget constraint $\sum_{j=0}^{N} |S_j^k| = \frac{n}{K}$ using bisection. This search is monotone and incurs $O(\log n)$ overhead, as detailed in Section B.1.

**Decentralized sampling.** Rather than relying on a master node to dispatch the assigned shards $\{S^k\}_{k=1}^K$, we use a shared random seed to reproduce the sampling process across $K$ nodes, and node $k$ prefetches only its local data shard $S^k$. When the stratum size permits, this avoids duplicate assignments, reduces redundant computation, and increases the number of distinct samples processed across nodes. This is consistent with the common practice of distributed samplers (Haddadpour et al., 2019; Yun et al., 2022). When fully disjoint assignments are infeasible, overlap across nodes is allowed to avoid undersampling from small strata, which would otherwise weaken the local approximation guarantee. This design eliminates central bandwidth bottlenecks and single points of failure.

**Hyper parameters of training.** The per-GPU batch sizes for these datasets are 16, 16, 128, 128, 16, 16, which are kept constant as the number of logical nodes varies. One training step is counted after every GPU processes one mini-batch of the specified per-GPU batch size. We use the AdamW optimizer (Loshchilov & Hutter, 2019) by default with `lr=1e-5` for OpenWebMath-Pro and Jiuzhang3.0, `lr=1e-3, wd=5e-4` for others.

## D Hyperparameter Analysis and Broader Empirical Validation

This section studies sensitivity to $C_{\texttt{Lip}}$, the exponential partition base, and reconstruction frequency, and presents ablations on sample overlap and stratum reweighting. We also compare Coreset with greedy partitioning baselines and report LLM training curves and evaluations across node counts, compute heterogeneity, communication failures, synchronization strategies, architectures, optimizers, label noise, and additional synchronization intervals.

**Lipschitz-related hyperparameter $C_{\texttt{Lip}}$.** As mentioned in Section 3.1, the Lipschitz-related hyperparameter $C_{\texttt{Lip}} = MR + LR^2$ models the worst-case loss deviation in the local parameter region, and further influences the relative sampling ratios across loss strata. When losses during training are reused, Remark 3.2 shows that their staleness is absorbed into the same term up to a constant factor. The individual components $M$, $L$, and $R$ appear only in the theoretical analysis, so we treat $C_{\texttt{Lip}}$ as an integrated tunable term. We conduct sensitivity analysis on CIFAR-10 and Tiny-ImageNet. The results are shown in Table 2. The experimental results suggest that the test accuracy is relatively stable across a wide range of $C_{\texttt{Lip}}$ values, with only a slight decrease when $C_{\texttt{Lip}}$ becomes overly large.

Table 2: Sensitivity to the Lipschitz-related hyperparameter $C_{\texttt{Lip}}$ under synchronization interval $H = 10$. Test accuracy (%) and standard deviation are reported over three independent runs. The two rows correspond to CIFAR-10 with ResNet-18 and Tiny-ImageNet with ResNet-50, respectively.

| Dataset | 0.01 | 0.05 | 0.1 | 0.2 | 0.4 | 0.6 | 0.8 | 1.0 | 3.0 | 5.0 | 10.0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| CIFAR-10 | 94.6±0.1 | 94.6±0.1 | 94.5±0.1 | 94.6±0.1 | 94.2±0.2 | 94.3±0.1 | 94.2±0.1 | 94.0±0.2 | 93.7±0.1 | 93.6±0.1 | 93.0±0.1 |
| Tiny-ImageNet | 62.1±0.1 | 62.1±0.2 | 62.5±0.1 | 62.2±0.2 | 62.7±0.2 | 62.3±0.1 | 62.5±0.2 | 62.2±0.1 | 62.8±0.1 | 62.1±0.1 | 61.9±0.1 |

Table 3: Sensitivity to the exponential partition base on CIFAR-10 with ResNet-18 under synchronization interval $H = 10$. Test accuracy (%) and standard deviation are reported over three independent runs.

| Base | 1.2 | 1.4 | 1.6 | 1.8 | 2.0 | 3.0 | 5.0 | 10.0 | Random |
|---|---|---|---|---|---|---|---|---|---|
| #Strata | 61 | 34 | 25 | 20 | 17 | 11 | 8 | 6 | 1 |
| Accuracy | 95.1±0.1 | 94.9±0.1 | 94.7±0.1 | 94.5±0.1 | 94.6±0.1 | 94.4±0.1 | 94.1±0.1 | 94.0±0.2 | 93.8±0.1 |

Table 4: Effect of local objective reconstruction frequency on CIFAR-10 with ResNet-18 under synchronization interval $H = 10$. The total training budget is fixed at 60 epochs of full-data training. Here, $\rho$ denotes the per-epoch sampled data ratio, and the number in parentheses is the corresponding number of reconstructions. We report test accuracy (%) and standard deviation over three runs, accuracy gain over Random, and iteration speedup to the same training loss.

| $\rho$ (#Recon.) | 0.1 (600) | 0.2 (300) | 0.3 (200) | 0.4 (150) | 0.5 (120) | 0.6 (100) | 0.8 (75) |
|---|---|---|---|---|---|---|---|
| Random Acc. | 94.0±0.1 | 94.2±0.1 | 94.3±0.2 | 94.0±0.1 | 94.2±0.1 | 94.3±0.1 | 94.2±0.1 |
| Coreset Acc. | 94.5±0.1 | 94.6±0.1 | 94.6±0.1 | 95.0±0.1 | 94.6±0.1 | 94.8±0.1 | 94.4±0.1 |
| Acc. Gain | +0.6 | +0.5 | +0.3 | +0.9 | +0.4 | +0.5 | +0.2 |
| Speedup | 1.35× | 1.43× | 1.42× | 1.40× | 1.41× | 1.27× | 1.14× |

This indicates that the proposed coreset construction is robust to the choice of this hyperparameter, and we use $C_{\texttt{Lip}} = 0.1$ across our experiments without task-specific tuning.

**Exponential partition base.** By default, we use an exponential partition with $\texttt{Base} = 2.0$ to define the loss strata in Eqs. (3, 4) across our experiments. Nevertheless, our theoretical results remain valid for any $\texttt{Base} > 1$. Here, we train CIFAR-10 with various choices of Base and the results are shown in Table 3. The number of loss strata is $N + 1 = \lceil \log_{\texttt{Base}} n \rceil + 1$, with $n = 50000$ training samples for CIFAR-10. The results suggest that test accuracy remains robust across different Base values, with positive gains over Random for all tested choices. When Base becomes too large, e.g., $\texttt{Base} \geq 5.0$, the number of strata becomes small, which weakens the granularity of loss stratification and reduces the accuracy gain over Random.

**Local objectives reconstruction frequency.** As mentioned in Section 3.1, we use $\rho$ to control the per-epoch sampled data ratio and, equivalently, the frequency of local-objective reconstruction under fixed training budgets. We empirically evaluate the effect of $\rho$ on CIFAR-10 and report the results in Table 4. The total iterations are aligned to 60 epochs of full-data training, so the number of reconstructions is inversely proportional to $\rho$. Across the tested $\rho$ values, Coreset improves test accuracy by at least $0.2\%$ over Random and achieves $1.14\times$–$1.43\times$ iteration speedups. For the tested values from $\rho = 0.2$ to $0.5$, the speedup is $1.40\times$–$1.43\times$, with accuracy gains up to $0.9\%$. These results suggest that the proposed construction is relatively robust to the choice of $\rho$. In our default setting, we choose $\rho$ to obtain 200 reconstructions for vision tasks and 20 reconstructions for LLM tasks.

**Greedy partitioning baselines.** We also compare with the greedy partitioning baselines Greedy-Facility and Greedy-GraphCut (He & Dube, 2023). They construct class-wise partitions with submodular greedy selection so that each worker receives a balanced and representative shard. Greedy-Facility favors representative coverage of each class, while Greedy-GraphCut maximizes the summed pairwise similarities between selected samples and the remaining samples in the same class. Both variants rely on pairwise similarity scores and repeatedly evaluate candidate additions, making each fresh construction more expensive than random partitioning or our loss-stratified con-

struction. Following the one-time partitioning protocol in RCD-SGD (He & Dube, 2023), the data shards of Greedy-Facility and Greedy-GraphCut are constructed once before training and reused across all epochs, whereas Random and Coreset perform lightweight epoch-wise reconstruction. The measured runtime breakdown is summarized in Table 5. Coreset incurs a negligible overhead of $0.4\%$ compared with Random, while the greedy baselines require substantially more time for construction. Such reconstruction costs can offset the savings from less frequent synchronization, introducing a practical challenge for epoch-wise greedy reconstruction in this setting.

Table 5: Measured wall-clock time for the greedy comparison on CIFAR-10 with ResNet-18 and synchronization interval $H = 10$. "Feat. ext." and "Greedy sel." denote feature extraction and greedy selection, respectively.

| Method | Feat. ext. (s) | Greedy sel. (s) | Total (s; vs. Random) |
|---|---|---|---|
| Random | – | – | 1966 (+0.0%) |
| Coreset | – | – | 1973 (+0.4%) |
| Greedy-Facility | 17.48 | 3446 | 5397 (+174.5%) |
| Greedy-GraphCut | 17.05 | 755 | 2737 (+39.2%) |

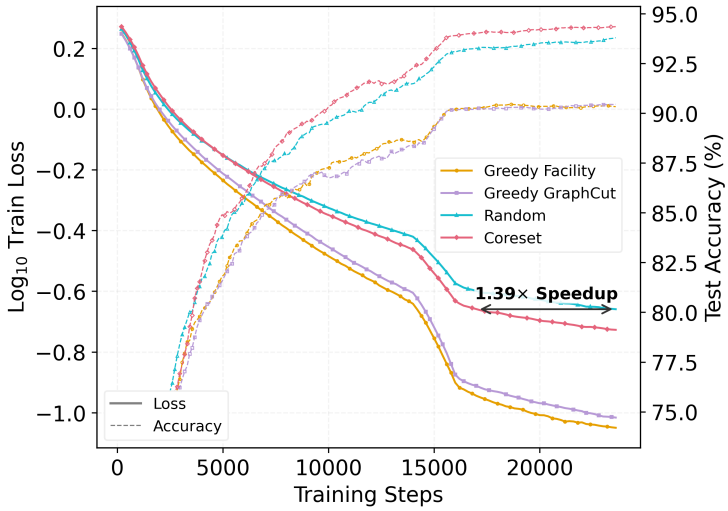



Figure 6: Comparison with greedy partitioning baselines on CIFAR-10 with ResNet-18 and synchronization interval $H = 10$. Solid and dashed lines denote training loss and test accuracy, respectively. Greedy Facility and Greedy GraphCut reduce training loss quickly but saturate at lower test accuracy.

Figure 6 compares the optimization and test performance of the partitioning baselines. The mini-batch order remains stochastic for all methods. Greedy-Facility and Greedy-GraphCut reduce training loss rapidly, but their test accuracy remains lower than Random and Coreset. Coreset achieves $1.39\times$ convergence speedup over Random and the highest test accuracy among the compared methods. This gap highlights the importance of evaluating a sharding strategy through both optimization and test performance. The static greedy baselines optimize submodular utilities for coverage or similarity, which can be effective for reducing training loss on the assigned shards. However, these utilities do not directly constrain whether each local objective remains close to the global objective during training. Coreset instead constructs weighted shards so that each local objective approximates the global objective, while retaining lightweight reconstruction across epochs.

**Ablation on sample overlap across nodes.** We compare how independent sampling across nodes affects data coverage and training performance under the same sampling budget on each node. All variants sample without replacement within each node. For Random, independent assignment lets each node independently draw its own subset, and samples may overlap across nodes. Partition assignment first draws a global subset and then partitions it across nodes, which is the default data-parallel sampling in practice (Li et al., 2020). Similarly, for Coreset, independent assignment lets each node independently sample from each stratum according to the stratum-wise budget in

Table 6: Ablation on sample overlap across nodes on Tiny-ImageNet with ResNet-50 and synchronization interval $H = 10$. Results are averaged over three independent runs. Unique coverage denotes the percentage of distinct samples among all samples assigned across nodes (averaged over all epochs). Acc. Gain and Iteration Speedup are measured against Random with partition assignment.

| Method | Assignment | Unique Cov. (%) | Test Acc. (%) | Final Loss | Acc. Gain (%) | Iteration Speedup |
|---|---|---|---|---|---|---|
| Random | Independent | 87.8 | 61.2±0.2 | 0.81±0.01 | −0.7 | 0.93× |
| Random | Partition | 100.0 | 61.9±0.1 | 0.77±0.02 | +0.0 | 1.00× |
| Coreset | Independent | 78.5 | 62.4±0.2 | 0.64±0.02 | +0.5 | 1.12× |
| Coreset | Quasi non-overlap | 88.2 | 62.9±0.2 | 0.61±0.01 | +1.0 | 1.18× |

Table 7: Ablation on weighted stratification on Tiny-ImageNet with ResNet-50 and synchronization interval $H = 10$. Results are averaged over three independent runs. Stratified Random uses the same loss strata as Coreset, samples from each stratum proportionally to its size, and assigns uniform weights to selected samples. Acc. Gain and Iteration Speedup are measured against Random.

| Method | Allocation | Weighting | Test Acc. (%) | Final Loss | Acc. Gain (%) | Iteration Speedup |
|---|---|---|---|---|---|---|
| Random | Uniform | Uniform | 61.9±0.1 | 0.77±0.02 | +0.0 | 1.00× |
| Stratified Random | Proportional | Uniform | 62.1±0.1 | 0.73±0.01 | +0.2 | 1.06× |
| Coreset | Serfling-based | Serfling-based | 62.9±0.2 | 0.61±0.01 | +1.0 | 1.18× |

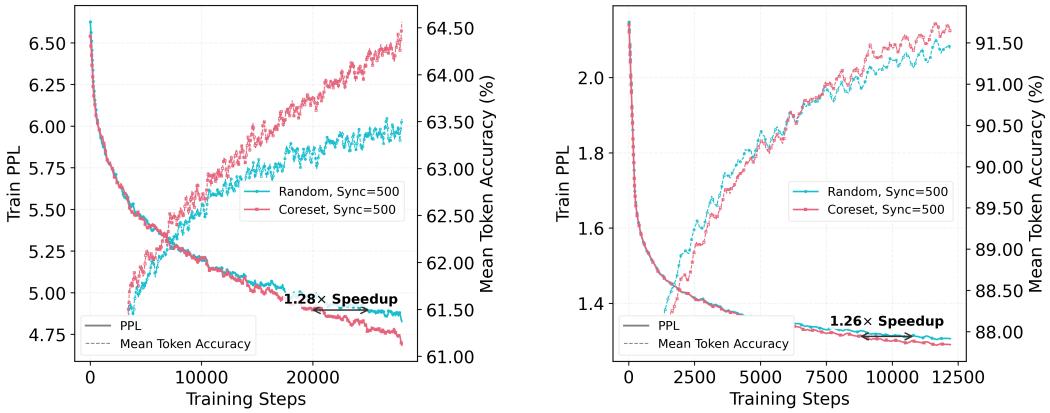
Lemma 3.1. The quasi non-overlap assignment is our default implementation in Algorithm 1. It avoids duplicate assignments within each stratum whenever the stratum size permits, and allows overlap only for small strata to bound the local approximation error. The experimental results in Table 6 show that independent sampling reduces unique coverage and degrades training performance under the matched sampling budget.

**Ablation on weighted stratification.** The allocation rule derived from Serfling's inequality controls the approximation error across strata independent from the worst-case loss, while the corresponding weights preserve unbiasedness. To isolate the contribution of the derived allocation and weighting, we compare with an unweighted variant that samples from each stratum proportionally to its size. This variant, denoted as "Stratified Random", can be viewed as a stratified version of random partitioning (Random). We conduct this ablation on Tiny-ImageNet with ResNet-50 and synchronization interval $H = 10$. The results in Table 7 show that unweighted stratification slightly improves over Random, with a $+0.2\%$ accuracy gain and $1.06\times$ iteration speedup, but remains below the proposed Coreset construction. Coreset further increases the gain to $+1.0\%$ and $1.18\times$, demonstrating the additional benefit of the derived allocation and weighting. The Serfling-based construction jointly determines the stratum-wise allocation and stratum weights, further balancing the approximation error across strata and improving both training loss and test accuracy.

**Training curves of LLM math continual pretraining.** Figure 7 shows that Coreset reaches the same perplexity with fewer iterations than Random on both datasets, achieving $1.28\times$ and $1.26\times$ iteration speedups on OpenWebMath-Pro and Jiuzhang3.0.

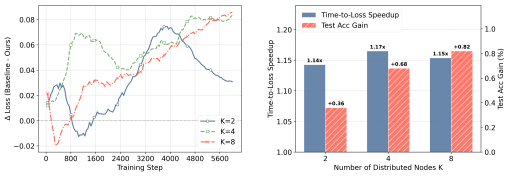
**Numbers of distributed nodes $K$.** By default, we use $K = 8$ nodes across our experiments. To further compare the performance under different node scales, we train ResNet-50 on Tiny-ImageNet with varying numbers of nodes $K \in \{2, 4, 8\}$. The results are shown in Figure 8. We keep the per-GPU batch size, total GPU resources, and total iteration budget fixed across different $K$. For all compared values of $K$, Coreset achieves a lower training loss than Random as training proceeds. Across $K \in \{2, 4, 8\}$, Coreset achieves $1.14\times$–$1.17\times$ iteration speedup and improves test accuracy by $0.36\%$–$0.82\%$ over Random.

**Heterogeneous compute capacity.** Beyond the homogeneous setting where all nodes have identical compute capacity, we further evaluate the performance under compute heterogeneity. We keep the total GPU count fixed, and assign different numbers of GPUs to each logical node specified by an imbalance ratio. Here, we use $K = 3$ logical nodes and a total of 8 GPUs for CIFAR-10 training. The results are shown in Figure 9. Across the three compared imbalance ratios, Coreset achieves a



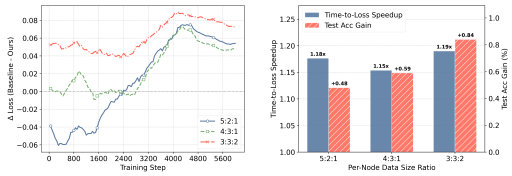

(a) OpenWebMath-Pro. (b) Jiuzhang3.0.

Figure 7: LLM math continual pretraining results using TinyLlama-1.1B with synchronization interval $H = 500$. Solid curves show training perplexity (PPL), and dashed curves show mean token accuracy.




(a) Loss advantage. (b) Speedup and accuracy gain.

Figure 8: Effect of the number of distributed nodes $K$ on Tiny-ImageNet with ResNet-50 and synchronization interval $H = 20$. (a) Train loss advantage over Random. (b) Iteration speedup to the same loss and accuracy gain over Random.




(a) Loss advantage. (b) Speedup and accuracy gain.

Figure 9: Performance under heterogeneous computing capability on CIFAR-10 with ResNet-18, with node capacities denoted by ratios such as 5:2:1. (a) Train loss advantage. (b) Speedup to the same loss and accuracy gain.

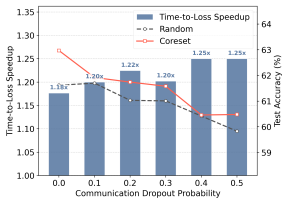
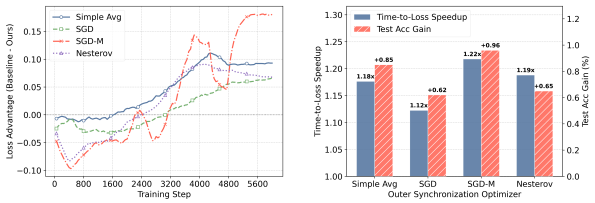



Figure 10: Random communication failures on Tiny-ImageNet with ResNet-50 and synchronization interval $H = 10$. We report test accuracy and iteration speedup to the same loss.




(a) Loss advantage. (b) Speedup and accuracy gain.

Figure 11: Synchronization strategies on Tiny-ImageNet with ResNet-50 ($H = 10$), including simple averaging and optimizer-based updates. (a) Train loss advantage. (b) Iteration speedup to the same loss and test accuracy gain over Random.

lower training loss than Random in the later training phase. It also achieves $1.15\times$–$1.19\times$ iteration speedup and $0.48\%$–$0.84\%$ test accuracy gains over Random.

**Random communication failures.** We further evaluate whether the proposed Coreset construction remains effective under unstable network conditions with random communication failures. To this end, we simulate communication failures during parameter synchronization with a specified dropout probability. The results of Tiny-ImageNet training are shown in Figure 10. As the dropout probabil-

Table 8: Additional results across model architectures and optimizers on CIFAR-10 with synchronization interval $H = 10$, where all models are trained from scratch. We report test accuracy (%) and standard deviation over three independent runs, accuracy gain over Random, and iteration speedup to the same training loss.

| Architecture | Optimizer | Random Acc. | Coreset Acc. | Acc. Gain | Speedup |
|---|---|---|---|---|---|
| ViT-B/16 | SGD-M | 58.7±0.4 | 61.0±0.5 | +2.3 | 1.30× |
| ViT-B/16 | LARS | 66.4±0.1 | 67.2±0.2 | +0.8 | 1.33× |
| ViT-B/16 | LAMB | 75.5±0.2 | 77.3±0.1 | +1.8 | 1.30× |
| ViT-B/16 | Lion | 67.5±0.3 | 69.0±0.2 | +1.5 | 1.28× |
| Swin-T | SGD-M | 64.7±0.4 | 67.0±0.4 | +2.2 | 1.28× |
| Swin-T | LARS | 59.7±0.3 | 61.4±0.3 | +1.7 | 1.22× |
| Swin-T | LAMB | 79.7±0.1 | 81.0±0.2 | +1.3 | 1.30× |
| Swin-T | Lion | 85.5±0.1 | 87.4±0.2 | +2.0 | 1.33× |

Table 9: Performance under label noise on CIFAR-10 with ResNet-18 and synchronization interval $H = 10$. We report test accuracy (%) and standard deviation over three independent runs, accuracy gain over Random, and iteration speedup to the same training loss.

| Corruption ratio | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 |
|---|---|---|---|---|---|
| Random Acc. | 92.7±0.1 | 91.1±0.0 | 90.0±0.1 | 88.7±0.1 | 85.8±0.2 |
| Coreset Acc. | 93.0±0.1 | 91.3±0.1 | 90.5±0.1 | 89.2±0.1 | 86.8±0.2 |
| Acc. Gain | +0.3 | +0.2 | +0.6 | +0.4 | +1.0 |
| Speedup | 1.40× | 1.22× | 1.18× | 1.15× | 1.31× |

ity increases from $0.0$ to $0.5$, both methods experience a decrease in test accuracy. Coreset achieves higher or comparable test accuracy across all dropout probabilities, with gains up to about $0.8\%$ over Random. It also maintains $1.18\times$–$1.25\times$ iteration speedup under random communication failures.

**Synchronization strategies.** By default, our experiments use simple parameter averaging at synchronization. Since data sharding is orthogonal to the choice of synchronization rule, we further examine the compatibility of Coreset with optimizer-based synchronization strategies, where the accumulated parameter updates from each node are aggregated and applied through an outer optimizer step (Douillard et al., 2024). We use momentum $0.9$ for SGD with momentum (SGD-M) and Nesterov momentum (Sutskever et al., 2013). A constant learning rate of $0.7$ is adopted for these outer optimizers following (Douillard et al., 2024). The results on Tiny-ImageNet training are shown in Figure 11. Figure 11 shows that the advantage becomes more pronounced later in training, with $1.12\times$–$1.22\times$ iteration speedups and $0.62\%$–$0.96\%$ test accuracy gains across the compared synchronization strategies. A possible explanation is that smaller discrepancies among local objectives enable more consistent updates for outer momentum accumulation. These results show that the gains of Coreset persist across both simple averaging and optimizer-based synchronization.

**Additional model architectures and optimizers.** For all entries in Table 8, we use batch size $1024$, synchronization interval $H = 10$, and simple averaging for parameter synchronization. The optimizer-specific learning rates, weight decay values, and momentum settings are kept fixed between Random and Coreset. All models are trained from scratch, and each row compares Random and Coreset under the same architecture, optimizer, and synchronization setting. Across all tested combinations, Coreset consistently reduces the number of iterations needed to reach the same training loss, with speedups from $1.22\times$ to $1.33\times$. At the same training budget, Coreset also improves test accuracy by $0.8\%$–$2.3\%$, suggesting that the weighted data shards improve both convergence speed and final test accuracy.

**Performance under label noise.** We evaluate ResNet-18 on CIFAR-10 with label corruption ratios ranging from $10\%$ to $50\%$, where randomly selected training labels are replaced with incorrect class labels. As shown in Table 9, Coreset improves test accuracy over Random across all corruption ratios, with a gain of up to $1.0\%$ at $50\%$ corruption, while maintaining iteration speedups of at least $1.15\times$. A possible explanation is that label corruption can increase variation in per-sample losses and exacerbate discrepancies among local objectives under Random partitioning. By constructing

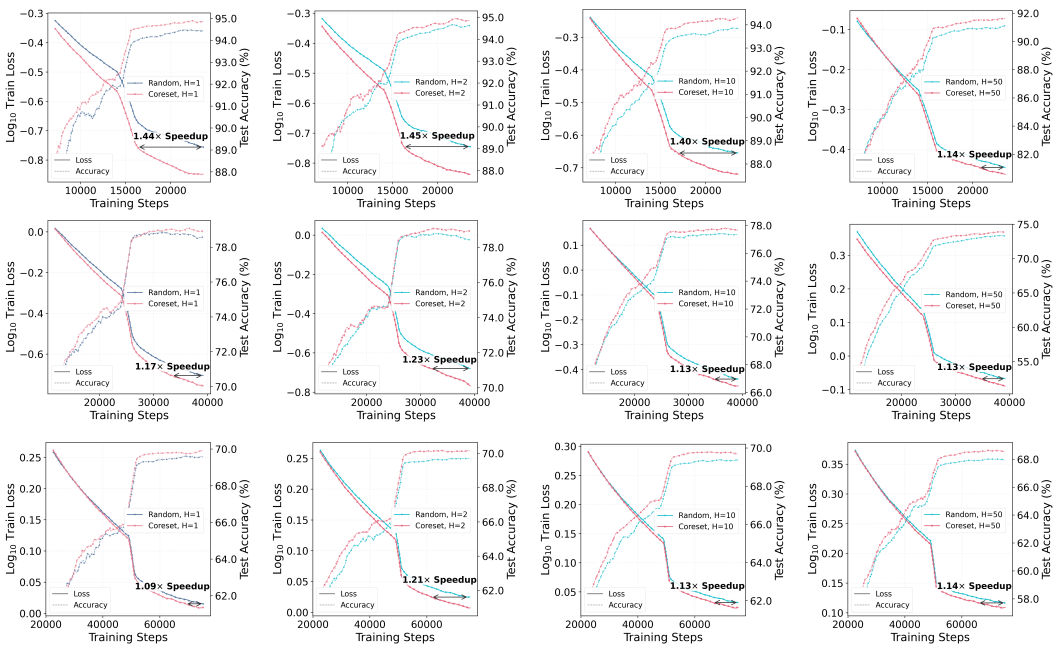



Figure 12: Additional convergence speedup and test accuracy results beyond the interval values shown in Figure 5. Rows correspond to CIFAR-10 with ResNet-18, CIFAR-100 with ResNet-50, and ImageNet-1K with ResNet-50. Columns correspond to $H = 1$, $H = 2$, $H = 10$, and $H = 50$, respectively.

weighted shards whose local objectives approximate the global objective across loss strata, Coreset may mitigate this effect, consistent with the local objective approximation guarantee in Lemma 3.1 and the lower cross-node objective discrepancy observed in Figure 3c.

**Additional results on more synchronization intervals.** We provide additional results for synchronization intervals not included in Figure 5. The results in Figure 12 further show that Coreset maintains consistent iteration speedups and test accuracy advantages across both small and large synchronization intervals.