SONU: COMMUNICATION-EFFICIENT ORTHONORMALIZED UPDATES VIA SKETCHING

Anonymous authors
Paper under double-blind review

ABSTRACT

As Large Language Model (LLM) training scales across distributed systems, gradient synchronization can incur substantial communication overhead, particularly in bandwidth-constrained settings. Gradient compression reduces this cost by communicating compact approximations of the gradient. Existing approaches, including sparsification and quantization, are typically designed to preserve the gradient under entrywise or Euclidean reconstruction criteria. These criteria, despite effective in coordinate-wise preconditioning optimizers, need not preserve the information used by matrix-orthonormalized optimizers such as Muon, whose update depends on the gradient's spectral geometry rather than its individual entries. Such discrepancy makes preserving the geometry an ideal target of the gradient compressor. We propose Sketched Orthonormalized Updates (SONU), a communication-efficient optimizer built on two-sided gradient sketching that facilitates synchronizing gradients in a low-dimensional space. SONU forms a generalized Nyström surrogate of the gradient from the sketches, which coincides with the dense gradient up to the sketched subspaces, and extracts the paired singular vectors without dense orthonormalization. To prevent a fixed complement from being persistently omitted, SONU maintains an evolving sketch, in which the sketching subspace is updated with the orthonormalized directions and are gradually transported across iterations. Empirically, on chain-of-thought, agentic function-calling, and instruction-following tasks, SONU attains lower perplexity per communicated byte than the other communication-efficient orthonormalized update algorithms. Furthermore, SONU matches the perplexity trajectory of Dion (Ahn et al., 2025), an efficient Muon variant which synchronizes the full gradients before applying orthonormalized updates.

1 INTRODUCTION

Distributed training can incur substantial gradient synchronization cost when workers communicate over bandwidth-constrained links (Austin et al., 2025; Huang et al., 2025). A large body of work therefore compress gradients using sparsification (Wang et al., 2023; Richtárik et al., 2022; Peng et al., 2026), quantization (Lu et al., 2022), or low-dimensional representations (Vogels et al., 2019; Song et al., 2023; Chen et al., 2025). However, existing gradient compression approaches are typically designed to preserve the gradient under entrywise or Euclidean reconstruction error. This objective can be misaligned with more recent matrix-orthonormalized optimizers such as Muon (Jordan et al., 2024) and Dion (Ahn et al., 2025).

Unlike the coordinate-wise preconditioning of AdamW (Loshchilov & Hutter, 2017), matrix orthonormalization updates leverage the spectral geometry of the momentum matrix. For a momentum matrix with singular value decomposition $M = U\Sigma V^\top$, the idealized orthonormalized direction is $UV^\top$, retaining paired left and right singular directions while normalizing their singular values. Consequently, the information consumed by these optimizers is not simply the magnitudes of individual gradient entries. This motivates designing the communicated information around the spectral geometry required by the optimizer, rather than treating compression as an optimizer-agnostic preprocessing step.

Matrix sketching offers an appealing alternative to the existing gradient compressors because it communicates low-dimensional linear observations of a matrix rather than perturbing its entries

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.

independently (Liberty, 2013; Chen et al., 2025). For orthonormalized updates, two-sided sketches can form the image of the matrix in certain row and column subspaces, providing direct information about its spectral geometry.

Despite the promise, combining sketching with orthonormalized updates requires resolving three closely related challenges. **First**, preserving paired spectral geometry: Information about the left and right subspaces must retain their coupling as in the canonical orthonormalized updates. Naively sketching the two subspaces independently is insufficient to retain this coupling. **Second**, avoiding persistently omitted complement subspaces: A low-dimensional observation inevitably leaves some gradient components unobserved. If the sketching subspaces remain fixed, the same complementary components can be invisible throughout training, which might hurt learning meaningful directions in optimization. **Third**, maintaining momentum in varying subspaces: Varying the sketch subspaces addresses the persistent omission issue but changes the bases in which momentum is represented. Directly accumulating buffers from different subspaces can mix incompatible representations. Overall, the algorithm design of a communication-efficient orthonormalized update must connect the geometry of the single-step gradients with the accumulation of varying subspaces across iterations. These challenges lead to the central question of this work: *Can sketched gradient communication preserve enough spectral information for orthonormalized updates while maintaining communication efficiency and expressivity?*

We propose **Sketched Orthonormalized Updates (SONU)**, a communication-efficient optimizer whose bidirectional communication between the master and workers remains entirely low-dimensional. Workers synchronize two-sided gradient sketches, from which SONU constructs a generalized Nyström surrogate of the dense gradient. The surrogate preserves the coupling between the observed left and right geometry, providing the paired singular directions needed for an orthonormalized update. LoRA factor gradients (Hu et al., 2022) provide a practical realization of these two-sided sketches without materializing dense gradients on individual workers.

To avoid excluding a fixed subspace complement for using a static sketch throughout, SONU evolves its sketching subspace along the optimization trajectory. The sketching projectors are updated with the orthonormalized update, and change which gradient subspaces are observed in each iteration. Furthermore, SONU naturally extends from **gradient** sketching to maintaining a subspace-aligned **momentum** and performs orthonormalization on the momentum, which aligns with (Jordan et al., 2024; Ahn et al., 2025).

Our contributions are summarized as follows:

1. We propose SONU, an optimizer targeting communication-efficient orthonormalized updates through two-sided gradient sketches. A generalized Nyström construction recovers paired singular directions from the low-dimensional communicated sketches. Factoring the sketches and decomposing a small core then yields the surrogate's paired singular vectors without dense orthonormalization. SONU need not materialize or store dense gradients on individual workers, unlike error-feedback approaches (Gruntkowska et al., 2026).

2. Evolving sketches and subspace-aligned momentum allow SONU to explore new subspaces while retaining the gradient geometry accumulated across iterations. We further propose a residual accumulation mechanism that retains the dropped component in the inexact subspace alignment step. Ablation studies further demonstrate that each design component effectively contributes to SONU's empirical performance.

3. We evaluate SONU on chain-of-thought reasoning, agentic function-calling, and instruction-following tasks. Across the evaluated settings, SONU achieves lower perplexity per communicated byte than the compared communication-efficient orthonormalized-update methods, while also matching the perplexity trajectory of Dion (Ahn et al., 2025), which synchronizes dense gradients before forming its orthonormalized updates.

2 METHODOLOGY

In this section, we develop **Sketched Orthonormalized Updates (SONU)**, a communication-efficient optimizer that constructs orthonormalized updates from two-sided, low-dimensional gradient sketches. SONU constructs a generalized Nyström surrogate for the full gradient using the gradient sketches,

Algorithm 1 Sketched Orthonormalized Update (SONU)

Input: Worker data $\{\mathcal{D}_k\}_{k=1}^K$, training steps T, initial weight W, learning rate η, momentum β, switching interval S, sketch dimension d, evolution rate α, initialization scale γ.
Output: Updated W, P^L, P^R

```
Initialize sketching matrices P^L, P^R with Eq. (13).
Initialize projected momentum buffers M̄^L, M̄^R and residual buffers E^L, E^R to zero.
for t = 1, 2, ..., T do
    On worker node k ∈ [K]:
        Compute local two-sided gradient sketches S_k^L, S_k^R on a sampled minibatch B_k ⊂ D_k.
    On master node:
        Average local sketches to obtain S^L, S^R with Eq. (1).
        Align and update momentum buffers M̄^L, M̄^R with Eq. (10) and (11).
        Update alignment residuals E^L, E^R with Eq. (12).
        Compute and broadcast U, V from M̄^L, M̄^R with Eq. (8).
    On worker nodes:
        If ⌊t/S⌋ is even:                    Otherwise:
            Update P^L ← P^L − ηαU.              Update P^R ← P^R − ηαV.
    Progressive orthonormalization on P^L, P^R and base update on W with Eq. (14).
end for
```

computes paired singular directions, and extends to projected momentum under evolving sketching subspaces. The full pipeline is summarized in Algorithm 1.

2.1 Surrogate Gradient Orthonormalization with Two-sided Sketches

Consider a layer with weight matrix $W \in \mathbb{R}^{I\times J}$, and let $G := \frac{1}{K}\sum_{k=1}^K \nabla_W \mathcal{L}_k(W) = \nabla_W \mathcal{L}(W)$ denote its global gradient across K workers. SONU maintains a left sketching matrix $P^L \in \mathbb{R}^{I\times d}$ and a right sketching matrix $P^R \in \mathbb{R}^{J\times d}$, where $d \ll \min\{I, J\}$. Denote the local sketches as $S_k^L := \nabla_W \mathcal{L}_k(W)P^R$ and $S_k^R := \nabla_W \mathcal{L}_k(W)^\top P^L$. Rather than synchronizing G, the workers communicate the two low-dimensional sketches

$$S^L := \frac{1}{K}\sum_{k=1}^K \nabla_W \mathcal{L}_k(W)P^R = GP^R \in \mathbb{R}^{I\times d},\ S^R := \frac{1}{K}\sum_{k=1}^K \nabla_W \mathcal{L}_k(W)^\top P^L = G^\top P^L \in \mathbb{R}^{J\times d}. \quad (1)$$

The average recovers global gradient sketches by linearity. As a two sided sketching of G, S^L captures column-space information from G, while S^R captures row-space information. At this point, we consider a momentum-free Muon update, i.e. $UV^\top = \mathrm{mSign}(G)$, and defer the discussion on momentum to Section 2.3. Note that the columns of U and V are *paired*, i.e., the i-th column of U and the i-th column of V are associated with the same singular value of G. Consequently, separate information about the two-sided sketches of G does not meet such requirement and constructing a Muon-style orthonormalized update requires an extra step to identify the pairing between S^L and S^R.

This challenge motivates us to construct an informative surrogate $\hat{G} \in \mathbb{R}^{I\times J}$ jointly from S^L and S^R, followed by orthonormalization of $\hat{G}$. The surrogate $\hat{G} \in \mathbb{R}^{I\times J}$ should satisfy two properties: (i) *subspace alignment and cross-sketch consistency*, meaning that the surrogate's column and row spaces are contained in those of S^L and $(S^R)^\top$, respectively, and that $\hat{G}$ preserves the shared cross-sketch matrix, i.e., $(P^L)^\top \hat{G} P^R = (P^L)^\top S^L = (S^R)^\top P^R$; and (ii) *bounded to rank-d*, ensuring the surrogate remains compact and contains no components outside current sketch subspaces. Based on such motivation, we leverage the following generalized Nyström form (Nakatsukasa, 2020) as the surrogate:

$$\hat{G} := S^L((P^L)^\top S^L)^\dagger (S^R)^\top, \quad (2)$$

where $(\cdot)^\dagger$ denotes a pseudo-inverse of the matrix. One can easily verify that this surrogate is up to rank-d, and agrees with the partial row and column space of the full gradient as in (1).

Another key advantage of the gradient surrogate defined in (2) is that the *paired* singular vectors of $\hat{G}$ can be computed efficiently. Instead of applying SVD (or Newton–Schulz iterations) directly to the

full matrix $\hat{G}$, we can leverage the low-dimensional structure of the gradient sketches. Concretely, let $S^R = Q^R\Gamma^R$ and $S^L = Q^L\Gamma^L$ be QR factorizations of the gradient sketches, where $Q^R \in \mathbb{R}^{J\times d}$ and $Q^L \in \mathbb{R}^{I\times d}$ have orthonormal columns, and $\Gamma^R, \Gamma^L \in \mathbb{R}^{d\times d}$ are upper triangular. We define the small core matrix $H := \Gamma^L\big((P^L)^\top S^L\big)^\dagger(\Gamma^R)^\top$. Let $H = U_H\Sigma_H V_H^\top$ be the SVD of H. The SVD of $\hat{G}$ is then given directly by $\hat{G} = (Q^L U_H)\Sigma_H(Q^R V_H)^\top$, where $U := Q^L U_H$ and $V := Q^R V_H$ are the *paired* left and right singular vectors of $\hat{G}$, respectively. The paired U and V are transmitted to all workers, matching the downlink (master-to-worker) communication cost to the uplink cost, both operating strictly in low-dimensional spaces. Each worker then updates the base weight matrix via $W \leftarrow W - \eta UV^\top$. Overall, SONU avoids orthonormalizing the dense gradient surrogate to compute U and V. We present the workflow in this section in Algorithm 2 and view it as a base algorithm that we keep refining in the following sections.

Remark 1. SONU is formulated using specific sketching matrices P^L and P^R. In practice, these sketching operations can be implemented via auxiliary low-rank factors in a LoRA-style parameterization (P^L, P^R). Following the insight from (Hao et al., 2024; Wang et al., 2024; 2025), a standard backpropagation then yields $\nabla_{P^L}\mathcal{L} = GP^R = S^L$ and $\nabla_{P^R}\mathcal{L} = G^\top P^L = S^R$. Thus, both two-sided sketches are obtained within a single backward pass without materializing G on the worker side (see Appendix C for implementation details). Importantly, SONU applies preconditioning directly in the dense weight space and updates the base weights, distinguishing it from LoRA-based methods (Ghosh et al., 2026) that perform adaptive optimization on adapter parameters.

Remark 2. Like Dion (Ahn et al., 2025), the per-iteration update in SONU is low-rank and represented by the factor matrices U and V. Both methods accumulate these low-rank updates across training steps, enabling higher-rank cumulative changes in the weight space over time. Crucially, SONU differs in synchronizing only two low-dimensional gradient sketches, whereas Dion implementation with master-node held momentum requires transmitting the full gradient.

2.2 Enhancing Expressivity via Evolving Sketch

While the surrogate $\hat{G}$ in (2) adapts to the current gradient's row and column spaces, its expressivity is limited by the fixed sketching subspaces. Because P^L and P^R compress the dense gradient into low-dimensional subspaces, any components residing in the orthogonal complements, spanned by $(P^L)^\perp \in \mathbb{R}^{I\times(I-d)}$ and $(P^R)^\perp \in \mathbb{R}^{J\times(J-d)}$, is unobserved in that iteration. To demonstrate the persistent omission issue, we consider the following potentially full-rank gradient realization:

$$G_\Phi = S^L\left((P^L)^\top S^L\right)^\dagger (S^R)^\top + (P^L)^\perp\Phi\left((P^R)^\perp\right)^\top = \hat{G} + (P^L)^\perp\Phi\left((P^R)^\perp\right)^\top, \tag{3}$$

where Φ can be any matrix in $\mathbb{R}^{(I-d)\times(J-d)}$. Every realization in (3) satisfies the cross-sketch consistency $(P^L)^\top G_\Phi P^R = (P^L)^\top S^L = (S^R)^\top P^R$, because the additive component lies in the orthogonal complements $(P^L)^\perp\Phi\big((P^R)^\perp\big)^\top$ of the current sketching subspaces.

In Section 2.1, the sketching matrices P^L and P^R remain fixed throughout training. Consequently, Algorithm 2 repeatedly observes the gradient through the same row and column subspaces. Directions lying in the orthogonal complements of these fixed sketching subspaces can therefore remain underrepresented. Intuitively, even a highly accurate surrogate may keep "missing" the same components of the full gradient if the sketching matrices never evolve.

Remark 3. The two additive terms in (3) may (and are highly probable to) have overlapping column and row spaces, because S^L (*resp.* S^R) need not be orthogonal to $(P^L)^\perp$ (*resp.* $(P^R)^\perp$). Therefore, the orthonormalized update derived from $\hat{G}$ can still contain components in the complementary sketching subspaces, preserving some gradient information beyond the fixed sketching directions, and keeping the gradient expressive at a certain level.

Alternating updates on Sketching Matrices. To mitigate this limitation, we allow the sketching matrices themselves to evolve during training. Given an orthonormalized update direction $UV^\top$, we evolve the left sketching matrix along U and the update to the base weight W remains unchanged:

$$P^L \leftarrow P^L - \alpha\eta U, \qquad P^R \leftarrow P^R, \qquad W \leftarrow W - \eta UV^\top, \tag{4}$$

where the evolution rate $\alpha \in \mathbb{R}$ controls the scale of the per-round sketch evolution. Symmetrically, we evolve the right sketching matrix along V via

$$P^L \leftarrow P^L, \qquad P^R \leftarrow P^R - \alpha\eta V, \qquad W \leftarrow W - \eta UV^\top. \tag{5}$$

Because the sketching matrices are updated strictly using singular vectors, this evolution step incurs no additional communication overhead, keeping the downlink cost identical to Algorithm 2. Furthermore, updating with current singular vectors adaptively aligns the sketch subspaces toward the gradient subspace as optimization proceeds, enabling the model to capture more informative gradient directions in subsequent steps. Finally, we alternate between (4) and (5) every S rounds to ensure that both the left and right sketching subspaces evolve progressively throughout training. The overall procedure is summarized in Algorithm 3.

2.3 Incorporating Momentum in SONU

So far, Algorithms 2 and 3 exploit two-sided sketches of the current gradient G_t. Canonical orthonormalized updates, however, performs over a momentum matrix rather than the instantaneous gradient alone, where the dense momentum is a temporal accumulation of past gradients, i.e. $M_t = \beta M_{t-1} + G_t$. In particular, Muon's update is based on $\mathrm{mSign}(M_t)$. At first glance, this seems incompatible with our methodology, since the full momentum matrix is not directly available from the low-dimensional gradient sketches without explicitly materializing dense gradient surrogates. Nevertheless, the two-sided sketch structure is sufficient to maintain projected momentum and construct a momentum surrogate that remains faithful to the original Muon design.

We first consider the case that the sketching matrices are fixed (Alg. 2). Note that the linearly-transformed dense momentums follow the accumulation pattern

$$M_t^L := M_t P^R = (\beta M_{t-1} + G_t)P^R = \beta M_{t-1}^L + S_t^L, \quad M_t^R := M_t^\top P^L = \beta M_{t-1}^R + S_t^R. \tag{6}$$

Instead of orthonormalizing a dense momentum M_t, we maintain momentum **buffers** $\bar{M}^L$ and $\bar{M}^R$. Both buffers are initialized with all-zero-matrices, and are updated directly with the synchronized sketches. For fixed sketching matrices, the following recurrence can exactly recover (6).

$$\bar{M}_t^L := \beta \bar{M}_{t-1}^L + S_t^L = M_t^L = M_t P^R, \qquad \bar{M}_t^R := \beta \bar{M}_{t-1}^R + S_t^R = M_t^R = M_t^\top P^L. \tag{7}$$

Since $\bar{M}_t^L$ and $\bar{M}_t^R$ are two-sided sketches of the dense momentum, the generalized Nyström construction in (2) yields the momentum surrogate

$$\hat{M}_t = \bar{M}_t^L \left((P^L)^\top \bar{M}_t^L\right)^\dagger (\bar{M}_t^R)^\top, \tag{8}$$

SONU applies the same rank-d orthonormalization on $\hat{M}_t$ to retrieve the paired singular vectors. The workflow is presented in Alg. 4.

Next, we consider projected momentum under evolving sketching matrices. Once P_t^L and P_t^R evolve over training iterations, momentum accumulation becomes less straightforward because the previous momentum buffers are tied to the earlier sketching matrices:

$$M_t^L := M_t P_t^R = \beta M_{t-1} P_t^R + S_t^L = \beta M_{t-1}^L + \beta M_{t-1}(P_t^R - P_{t-1}^R) + S_t^L, \tag{9}$$

and similarly for $M_t^R = M_t^\top P_t^L$. The discrepancy arises because the desired momentum buffer is defined using the current sketching matrix, whereas the maintained buffer is inherited from the previous one. We therefore introduce an alignment mechanism that explicitly accounts for the change in sketching coordinates at each step that better approximates the evolution of the momentum buffer.

We compute alignment maps between consecutive sketching matrices and use them to transport the momentum buffer $\bar{M}_t^L$ into the current sketching coordinates before incorporating the new sketches.

$$\bar{M}_t^L := \beta \widetilde{M}_{t-1}^L + G_t^L P_t^R, \qquad \widetilde{M}_{t-1}^L = \bar{M}_{t-1}^L \left((P_{t-1}^R)^\top P_{t-1}^R\right)^\dagger (P_{t-1}^R)^\top P_t^R. \tag{10}$$

The current sketching matrix is expressed in the least-squares coordinates of its predecessor, thereby aligning the inherited momentum with the new sketching subspace. The transport is exact when the new sketching matrix lies in the column space of the previous one and is otherwise an approximation. Similar subspace-alignment techniques have also been explored in (Yen et al., 2024; Yu et al., 2026). The workflow is presented in Algorithm 5.

Alignment Residual Accumulation. Subspace alignment in (10) can discard historical momentum that falls outside the new sketching frame. To mitigate this information loss, we maintain ambient residual buffers $E^L, E^R \in \mathbb{R}^{I \times J}$ on the master node. Denote Moore-Penrose pseudo-inverse as

$(P_t^R)^+ := \left((P_t^R)^\top P_t^R\right)^{-1}(P_t^R)^\top$. At iteration t, we lift the historical momentum and combine it with E_{t-1}^L into an ambient estimate T_t^L, which is then projected into the current sketch subspace P_t^R:

$$T_{t-1}^L := \bar{M}_{t-1}^L (P_{t-1}^R)^+ + E_{t-1}^L, \qquad \widetilde{M}_{t-1}^L := T_{t-1}^L P_t^R. \tag{11}$$

The residual buffer E_t^L incorporates the mass not represented in $\widetilde{M}_{t-1}^L$ and then decays by the momentum factor β for the next iteration, whereas E_t^R receives no new residual and simply decays:

$$E_t^L := \beta\left(T_{t-1}^L - \widetilde{M}_{t-1}^L (P_t^R)^+\right) = \beta T_{t-1}^L\left(I - P_t^R (P_t^R)^+\right), \qquad E_t^R := \beta E_{t-1}^R. \tag{12}$$

By construction, $E_t^L P_t^R = 0$, indicating that the residual contains no components within the sketching subspace. As P^R evolves across iterations, previously unrepresented residuals can re-enter the sketch subspace, effectively recovering the historical momentum, while still keeping the orthonormalization step in the low-rank space. The update for $\bar{M}^R$ under an evolving P^L is symmetric. Note that the alignment and the residual accumulation steps are both performed on the master node, incurring no additional communication overhead. The residual accumulation remains an approximate correction to the projected momentum. It mitigates information loss during subspace alignment but cannot generally recover historical momentum components that were unobserved by past sketches.

2.4 Sketching Matrix Initialization and Progressive Orthonormalization

SVD-based sketching matrix initialization. The momentum surrogates defined in (8) involve pseudoinverses of small core matrices whose numerical stability depends on the geometry of the sketching matrices. Poorly conditioned matrices can distort the spectrum of the momentum and unduly amplify certain momentum directions. We therefore initialize the sketching matrices using the leading singular subspaces of the initial base weight while assigning a uniform scale across all directions. Let $W = U_W \Sigma_W V_W^\top$ be the SVD of the initial base weight. We set

$$P_0^L \leftarrow \gamma U_W[:, :d], \qquad P_0^R \leftarrow \gamma V_W[:, :d], \tag{13}$$

where γ is a scaling factor controlling the magnitude along each direction. While the magnitude of the sketching matrices does not directly affect the current momentum surrogate, its interaction with the sketch evolution rate α governs how rapidly the sketching matrices adapt over time. This design is inspired by PiSSA (Meng et al., 2024), but differs by enforcing a uniform singular geometry rather than retaining the original singular values, thereby improving the conditioning of subsequent pseudoinverse operations.

Progressive orthonormalization along the trajectory. Even with a well-conditioned initialization, repeated sketch updates can cause P^L and P^R to drift from an orthonormal geometry, particularly over long training horizons or when the evolution rate α is large. We therefore re-orthonormalize the sketching matrices after each update. Specifically, when updating P^L, let $Q := \mathrm{Orth}(P^L - \alpha\eta U)$ denote an orthonormal basis for the updated column space. The resulting update becomes

$$P^L \leftarrow \gamma Q, \qquad P^R \leftarrow P^R, \qquad W \leftarrow W - \eta U V^\top. \tag{14}$$

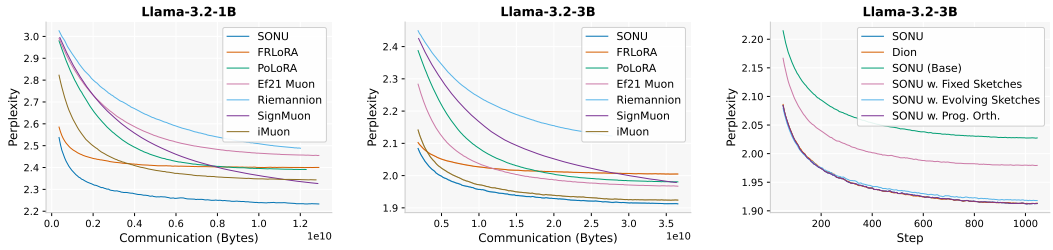
This projection preserves the evolving sketch subspace while maintaining a uniform scale across directions. This re-orthonormalization step is performed locally on the worker node using the already-transmitted U, which incurs no additional communication overhead. The full pipeline is presented in Algorithm 6.

3 Empirical Studies

In this section, we apply SONU to three representative language modeling tasks in LLM training, and demonstrate SONU robustly achieves the best performance among the communication-efficient baselines. We also showcase the effect of each algorithm design through a fine-grained ablation study.

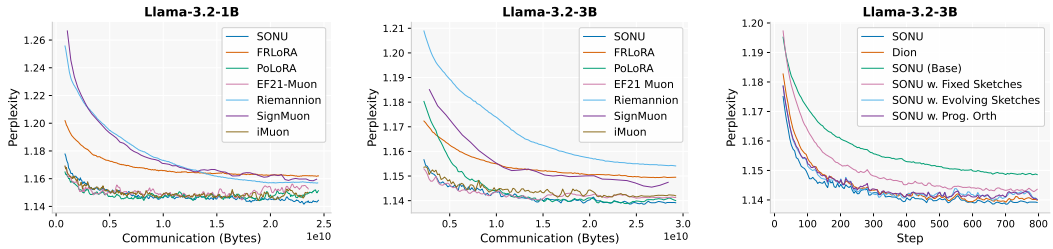
3.1 Experimental Setup

Datasets. Typical LLM language modeling tasks involves chain-of-thought training, agentic function calling, and instruction following. We build training datasets under supervised fine-tuning setting,



(a) (b) (c)

Figure 1: Perplexity trajectory on CoT training task validation set. Across model scales, SONU achieves the lowest perplexity among baselines and closely tracks Dion's trajectory. Ablation study in 1c demonstrates the gains from each component of SONU's design.



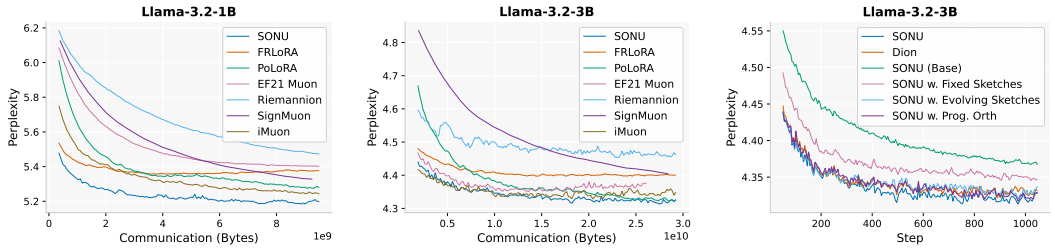
(a) (b) (c)

Figure 2: Perplexity trajectory on Agentic Function Calling task validation set. Across model scales, SONU matches or outperforms all baselines and closely tracks Dion's trajectory. Ablation study in 2c demonstrates the gains from each component of SONU's design.

and create *prompt-response* pairs for each sample. We choose public datasets: (1) MegaScience (Fan et al., 2025), where prompts are scientific question queries and responses are structured as Chain-of-Thought rationales; (2) APIGen (Liu et al., 2024), which pairs natural language queries and tool set definition with structured API executable responses formatted with JSON objects; and (3) OASST2 (OpenAssistant, 2023), where prompts include natural language instructions or questions and responses are expected outputs ranked highly by human contributors.

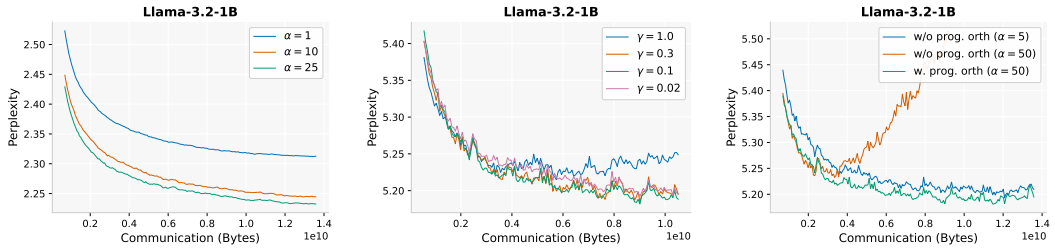
Baselines. We compare against three categories of baselines. The closest to our work is communication-compressed Muon, which keeps a Muon-type orthonormalized step while compressing what workers exchange. EF21-Muon (Gruntkowska et al., 2026) applies contractive compressors, such as Top-K sparsification, in both communication directions, with error feedback on the communication. SignMuon (Mishra et al., 2026) instead compresses after orthonormalization: each worker computes a Muon direction locally and transmits only its elementwise signs, which are aggregated by majority vote. The second category trains the full weights through LoRA factors by repeatedly merging them into the dense weight. FRLoRA (Yan et al., 2025) adds the residual of the aggregated factor product to the base weight at each global step and re-initializes the factors from the principal singular components of the pretrained weight, so that the accumulated dense update can exceed the adapter rank. The third category adapts Muon-type steps to the LoRA parameterization itself and updates only the adapters. PoLoRA (Ghosh et al., 2026) takes a preconditioned polar step on the factor gradients that accounts for the product structure of the adapter, while Riemannion (Bogachev et al., 2026) and iMuon (Li et al., 2026) formulate spectral-norm steps directly on the fixed-rank matrix manifold. As an uncompressed reference, we include Dion (Ahn et al., 2025) with master-node held momentum. It synchronizes the full dense gradients and then computes a low-rank orthonormalized update from the synchronized momentum. A more detailed discussion of the baselines can be found in Appendix. D.1.

Environments. We evaluate our approach on Llama-3.2-1B and Llama-3.2-3B base models (Grattafiori et al., 2024) in a distributed environment with 64 workers. Setting the sketching dimension to $d = 16$, we apply two-sided sketching to linear projection matrices within the



(a) (b) (c)

Figure 3: Perplexity trajectory on Instruction Following task validation set. Across model scales, SONU matches or outperforms all baselines and closely tracks Dion's trajectory. Ablation study in 3c demonstrates the gains from each component of SONU's design.



(a) (b) (c)

Figure 4: Perplexity trajectory under different update magnitude. 4a plots CoT Training tasks across varying α. 4b and 4c plot Instruction Following Task with varying γ and α respectively. 'prog. orth' in the legend stands for progressive orthonormalization. SONU's empirical performance benefits from larger effective update magnitude, and the orthonormalization step improves stability.

self-attention layers. This configuration achieves communication compression ratios of 2.03% for Llama-3.2-1B and 1.30% for Llama-3.2-3B. For all LoRA-based baselines, we use the same target modules and fix the LoRA rank to $r = 16$, matching SONU's per-step communication overhead.

3.2 Experimental Results

We evaluate language modeling performance using held-out perplexity, which directly reflects predictive likelihood under the next-token prediction objective. This is a common metric leveraged in related works (Ghosh et al., 2026; Ahn et al., 2025; Douillard et al., 2023). A lower perplexity value indicates better performance. Figures 1, 2, and 3 plot validation perplexity against per-worker cumulative communication (in bytes) for the three tasks. Across all six task–scale settings, SONU attains the lowest or tied-lowest final perplexity. In individual settings, a strong baseline can be competitive with SONU: PoLoRA matches it on agentic function calling and instruction following at 3B, while iMuon and EF21-Muon come close on CoT training and agentic function calling at 3B, respectively. However, no baseline remains competitive across tasks. On Agentic Function Calling dataset trained at 3B, SONU clearly improves over FRLoRA, Riemannion, SignMuon, and iMuon by a clear margin. On Instruction Following at 1B scale, SONU achieves a pronounced advantage against PoLoRA and EF21 Muon. Overall, these results demonstrate that SONU is competitive across the baseline families.

Moreover, SONU closely tracks Dion's perplexity curve throughout the training procedure. This finding is indeed surprising since it indicates that the exact dense gradients retained by Dion offer only marginal benefits to the downstream matrix-preconditioning optimizer. Despite replacing the dense gradient synchronization with an efficient sketched gradient communication, SONU retains nearly all empirical advantages of Dion. In certain configurations (e.g., 2c and Fig. 3c), SONU even slightly surpasses Dion at convergence. Note that Fig. 1c, 2c and 3c plot convergence as a function of

training steps, matching up with Dion's step-wise convergence rate confirms that SONU requires significantly less communication overhead to achieve an equivalent perplexity score.

To evaluate the individual contributions of our proposed algorithm designs, we conduct a fine-grained ablation study on Llama-3.2-3B across all three tasks (Fig. 1c, 2c and 3c); results for the 1B scale model are provided in Appendix D.3. While the magnitude of individual effects varies by task, a consistent trend emerges: each algorithmic component incrementally improves performance, ultimately enabling SONU to match the perplexity trajectory of Dion. First, the base SONU algorithm (Alg. 2) sets a performance that is comparable to other baselines, validating our surrogate gradient design. Applying momentum mechanism under fixed sketching matrices (Alg. 4) significantly accelerates the convergence, possibly by stabilizing the gradient surrogate in each round. Transitioning from fixed to evolving sketching matrices (Alg. 5) enhances the expressivity of the update directions, leading to substantial empirical gains across all tasks. In the Agentic Function Calling and Instruction Following tasks, Alg. 5 is already competitive with the performance of Dion. We also observe that uniform initialization and progressive orthonormalization Alg. 6 bridge the last-mile gap in the CoT Training task. As shown in Fig. 1c, Alg. 6 improves over Alg. 5 and ultimately achieves an empirical convergence rate matching Dion. Finally, incorporating the residual alignment mechanism (Alg. 1) further improves the performance in the Agentic Function Calling and Instruction Following tasks.

3.3 On the Magnitude of Update over Sketching Matrices

As established in our ablation studies, SONU significantly benefits from the evolving sketching matrix design. We further investigate how the update magnitude of the sketching matrices influences the optimization trajectory. Specifically, we analyze two pivotal hyperparameters of Alg. 6 on the Llama-3.2-1B scale: the update multiplier α in (4) and the orthonormalization scale factor γ in (13).

Fig. 4a illustrates a controlled experiment on the CoT task across varying α values. Since α is the sketching matrix's evolution rate, larger values correspond to more aggressive update. We observe that the perplexity consistently improves as α increases, with diminishing improvements for large values. Fig. 4b examines the influence of the sketching matrix scale γ. Smaller γ implies a reduced magnitude, thereby increasing the relative impact of each update. Fig. 4b indicates that smaller γ enhances the convergence rate, and the gain saturates at approximately $\gamma = 0.3$. These findings are consistent: SONU's performance benefits from large effective update applied to the sketching matrices, which aligns with our core design intuition, i.e. increasing expressivity of observed subspaces is essential for exploring the ambient gradient space.

However, the desired larger updates inherently risk optimization stability. We demonstrate that our progressive orthonormalization step is essential for maintaining stability under the aggressive update regime. We conduct a controlled sensitivity analysis on the instruction-following task trained on Llama-3.2-1B, where the scale of the sketching matrices γ is set at 0.1, and we only vary the application of progressive orthonormalization and the update multiplier α. As illustrated in Fig. 4c, while SONU's variant without orthonormalization performs competitively at low update scales ($\alpha = 5$), it becomes sharply divergent as the scale increases ($\alpha = 50$). In contrast, SONU with progressive orthonormalization remains performant even under these aggressive update magnitudes by delivering the lowest perplexity among the three configurations. These results suggest that the progressive orthonormalization step acts as a critical stabilizer. By preventing divergence under aggressive α, it reduces the necessity for exhaustive hyperparameter tuning, which makes SONU a robust optimizer for real-world deployment.

4 Conclusion

We introduced SONU, a communication-efficient optimizer that forms orthonormalized updates from two-sided gradient sketches. On chain-of-thought, agentic function-calling, and instruction-following tasks with Llama-3.2-1B and 3B, SONU communicates 1.3–2.0% of the dense gradient volume per step and attains lower perplexity per communicated byte than existing communication-efficient orthonormalized methods. It also matches the step-wise trajectory of Dion with server-held momentum. These results suggest that aligning the communicated information with the geometry the optimizer consumes is an effective principle for compressing matrix-preconditioned updates.

AI USE STATEMENT

In this work, we used generative AI tools for polishing the writing, related works retrieval, and partial code implementation. We have not used generative AI tools for drafting sections of the paper, generating synthetic datasets, or providing mathematical claims.

We have reviewed all AI-assisted work by checking the existence of the retrieved related works, and by verifying the LLM-generated codes. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We have disclosed all the details for reproducing the results, including the experiment setup and hyper-parameters in Section D.2. The code can be accessed at https://anonymous.4open.science/r/sonu_review-DEA5.

REFERENCES

Kwangjun Ahn, Byron Xu, Natalie Abreu, Ying Fan, Gagik Magakyan, Pratyusha Sharma, Zheng Zhan, and John Langford. Dion: Distributed orthonormalized updates. *arXiv preprint arXiv:2504.05295*, 2025.

Jacob Austin, Sholto Douglas, Roy Frostig, Anselm Levskaya, Charlie Chen, Sharad Vikram, Federico Lebron, Peter Choy, Vinay Ramasesh, Albert Webson, and Reiner Pope. How to scale your model. Online, 2025. Retrieved from https://jax-ml.github.io/scaling-book/.

Vladimir Bogachev, Vladimir Aletov, Alexander Molozhavenko, Denis Bobkov, Vera Soboleva, Aibek Alanov, and Maxim Rakhuba. Lora meets riemannion: Muon optimizer for parametrization-independent low-rank adapters. In *International Conference on Learning Representations*, volume 2026, pp. 82190–82215, 2026.

Zhijie Chen, Qiaobo Li, and Arindam Banerjee. Sketched adaptive distributed deep learning: A sharp convergence analysis. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Arthur Douillard, Qixuan Feng, Andrei A Rusu, Rachita Chhaparia, Yani Donchev, Adhiguna Kuncoro, Marc'Aurelio Ranzato, Arthur Szlam, and Jiajun Shen. Diloco: Distributed low-communication training of language models. *arXiv preprint arXiv:2311.08105*, 2023.

Run-Ze Fan, Zengzhi Wang, and Pengfei Liu. Megascience: Pushing the frontiers of post-training datasets for science reasoning. *arXiv preprint arXiv:2507.16812*, 2025. URL `https://arxiv.org/abs/2507.16812`.

Nikhil Ghosh, Tetiana Parshakova, and Robert M Gower. Polora: A preconditioned orthogonalized lora optimizer. *arXiv preprint arXiv:2607.17620*, 2026.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Kaja Gruntkowska, Alexander Gaponov, Zhirayr Tovmasyan, and Peter Richtárik. Error feedback for muon and friends. In *International Conference on Learning Representations*, volume 2026, pp. 94378–94449, 2026.

Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization. In *International Conference on Machine Learning*, pp. 1842–1850. PMLR, 2018.

Yongchang Hao, Yanshuai Cao, and Lili Mou. Flora: Low-rank adapters are secretly gradient compressors. *arXiv preprint arXiv:2402.03293*, 2024.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *International Conference on Learning Representations*, 2022.

Jinghan Huang, Hyungyo Kim, Nachuan Wang, Jaeyoung Kang, Hrishi Shah, Eun Kyung Lee, Minjia Zhang, Fan Lai, and Nam Sung Kim. Netzip: Algorithm/hardware co-design of in-network lossless compression for distributed large model training. In *Proceedings of the 58th IEEE/ACM International Symposium on Microarchitecture*, pp. 609–625, 2025.

Keller Jordan, Yuchen Jin, Vlado Boza, You Jiacheng, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. URL `https://kellerjordan.github.io/posts/muon/`.

Yibang Li, Bihari Lal Pandey, Ravi Sah, Andi Han, Cyrus Mostajeran, Pratik Jawanpuria, and Bamdev Mishra. Intrinsic muon: Spectral optimization on riemannian matrix manifolds. *arXiv preprint arXiv:2605.09238*, 2026.

Vladislav Lialin, Sherin Muckatira, Namrata Shivagunde, and Anna Rumshisky. Relora: High-rank training through low-rank updates. In *International Conference on Learning Representations*, volume 2024, pp. 49405–49421, 2024.

Edo Liberty. Simple and deterministic matrix sketching. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 581–588, 2013.

Hong Liu, Zhiyuan Li, David Hall, Percy Liang, and Tengyu Ma. Sophia: A scalable stochastic second-order optimizer for language model pre-training. *arXiv preprint arXiv:2305.14342*, 2023.

Zuxin Liu, Thai Hoang, Jianguo Zhang, Ming Zhu, Tian Lan, Shirley Kokane, Juntao Tan, Weiran Yao, Zhiwei Liu, Yihao Feng, et al. Apigen: Automated pipeline for generating verifiable and diverse function-calling datasets. *arXiv preprint arXiv:2406.18518*, 2024.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Yucheng Lu, Conglong Li, Minjia Zhang, Christopher De Sa, and Yuxiong He. Maximizing communication efficiency for large-scale training via 0/1 adam. *arXiv preprint arXiv:2202.06009*, 2022.

Fanxu Meng, Zhaohui Wang, and Muhan Zhang. Pissa: Principal singular values and singular vectors adaptation of large language models. *Advances in Neural Information Processing Systems*, 37: 121038–121072, 2024.

Neel Mishra, Kushagara Trivedi, and Pawan Kumar. Signmuon: Communication-efficient distributed muon optimization. *arXiv preprint arXiv:2605.16311*, 2026.

Yuji Nakatsukasa. Fast and stable randomized low-rank matrix approximation. *arXiv preprint arXiv:2009.11392*, 2020.

OpenAssistant. Openassistant conversations dataset v2 (oasst2). `https://huggingface.co/datasets/OpenAssistant/oasst2`, 11 2023.

Bowen Peng, Lizhang Chen, Baiyu Su, Jeffrey Quesnelle, Diederik Durk Kingma, and Qiang Liu. Decoupled momentum optimization. In *International Conference on Learning Representations*, volume 2026, pp. 14058–14083, 2026.

Peter Richtárik, Igor Sokolov, and Ilyas Fatkhullin. Ef21: A new, simpler, theoretically better, and practically faster error feedback. *Advances in Neural Information Processing Systems*, 34: 4384–4396, 2021.

Peter Richtárik, Igor Sokolov, Elnur Gasanov, Ilyas Fatkhullin, Zhize Li, and Eduard Gorbunov. 3pc: Three point compressors for communication-efficient distributed training and a better theory for lazy aggregation. In *International Conference on Machine Learning*, pp. 18596–18648. PMLR, 2022.

Daniel Rothchild, Ashwinee Panda, Enayat Ullah, Nikita Ivkin, Ion Stoica, Vladimir Braverman, Joseph Gonzalez, and Raman Arora. Fetchsgd: Communication-efficient federated learning with sketching. In *International Conference on Machine Learning*, pp. 8253–8265. PMLR, 2020.

Zhao Song, Yitan Wang, Zheng Yu, and Lichen Zhang. Sketching for first order method: efficient algorithm for low-bandwidth channel and vulnerability. In *International Conference on Machine Learning*, pp. 32365–32417. PMLR, 2023.

Thijs Vogels, Sai Praneeth Karimireddy, and Martin Jaggi. Powersgd: Practical low-rank gradient compression for distributed optimization. *Advances in Neural Information Processing Systems*, 32, 2019.

Nikhil Vyas, Depen Morwani, Rosie Zhao, Mujin Kwun, Itai Shapira, David Brandfonbrener, Lucas Janson, and Sham Kakade. Soap: Improving and stabilizing shampoo using adam. *arXiv preprint arXiv:2409.11321*, 2024.

Jue Wang, Yucheng Lu, Binhang Yuan, Beidi Chen, Percy Liang, Christopher De Sa, Christopher Re, and Ce Zhang. Cocktailsgd: Fine-tuning foundation models over 500mbps networks. In *International Conference on Machine Learning*, pp. 36058–36076. PMLR, 2023.

Shaowen Wang, Linxi Yu, and Jian Li. Lora-ga: Low-rank adaptation with gradient approximation. *Advances in Neural Information Processing Systems*, 37:54905–54931, 2024.

Zhengbo Wang, Jian Liang, Ran He, Zilei Wang, and Tieniu Tan. Lora-pro: Are low-rank adapters properly optimized? In *International Conference on Learning Representations*, volume 2025, pp. 93787–93808, 2025.

Yunlu Yan, Chun-Mei Feng, Wangmeng Zuo, Rick Siow Mong Goh, Yong Liu, and Lei Zhu. Federated residual low-rank adaptation of large language models. In *The Thirteenth International Conference on Learning Representations*, 2025.

Jui-Nan Yen, Si Si, Zhao Meng, Felix Yu, Sai Surya Duvvuri, Inderjit S Dhillon, Cho-Jui Hsieh, and Sanjiv Kumar. Lora done rite: Robust invariant transformation equilibration for lora optimization. *arXiv preprint arXiv:2410.20625*, 2024.

Xin Yu, Yujia Wang, Jinghui Chen, and Lingzhou Xue. Altlora: Towards better gradient approximation in low-rank adaptation with alternating projections. *Advances in Neural Information Processing Systems*, 38:37484–37522, 2026.

Algorithm 2 SONU Base

Input: Data shards $\{\mathcal{D}_k\}_{k=1}^K$, initial weights W, sketching matrices P^L, P^R, learning rate η, sketch dimension d
Output: Updated W

for $t = 1, 2, \ldots, T$ **do**
  for each worker $k = 1, \ldots, K$ in parallel **do**
    Sample minibatch $\mathcal{B}_k \subset \mathcal{D}_k$
    Compute local gradient sketches S_k^L, S_k^R
  end for
  Master averages local sketches $S^L = \frac{1}{K}\sum_k S_k^L$, $S^R = \frac{1}{K}\sum_k S_k^R$
  Construct $\hat{G} \leftarrow S^L((P^L)^\top S^L)^\dagger (S^R)^\top$
  Compute orthonormal factors $U, V = \mathrm{Orth}(\hat{G})$
  Update $W \leftarrow W - \eta UV^\top$
end for

A Related Works

Matrix-Preconditioning Based Optimizer. While AdamW (Loshchilov & Hutter, 2017) have dominated LLM training in the past decade, recent developments in optimization algorithms have shown potential to improve training efficiency (Liu et al., 2023; Jordan et al., 2024; Vyas et al., 2024; Gupta et al., 2018). These methods approximate second-order optimization with diagonal Hessian approximations or layerwise Kronecker-factored preconditioner. Recently, Muon (Jordan et al., 2024) proposes to use Polyak's momentum $M \leftarrow \beta M + G$, where G is the gradient and β is the momentum factor and update matrix parameters with the orthonormalized momentum. The orthonormalization can be defined by the matrix sign function $\mathrm{mSign}(M) := UV^\top$, where $U\Sigma V^\top$ is the SVD of M.

Gradient compression. Sparsification and quantization reduce communication by transmitting compressed gradient representations. Error-feedback methods compensate for compression errors across iterations: EF21 (Richtárik et al., 2021) develops an error-feedback mechanism for contractive compression, while 3PC (Richtárik et al., 2022) provides a broader compressor framework that includes EF21 as a special case. CocktailSGD (Wang et al., 2023) combines random sparsification, TopK sparsification, and quantization for communication-efficient fine-tuning of foundation models.

Sketching and bidirectional communication. Sketching provides another approach to gradient compression. Song et al. (2023) develop a sketch-and-desketch framework that communicates low-dimensional projections and reconstructs gradient information at the receiver. FetchSGD (Rothchild et al., 2020) aggregates Count Sketches, maintains momentum and error accumulation in sketch space on the server, and extracts sparse updates for broadcast. These designs also highlight the importance of downlink communication: averaging sparse worker updates with different supports can increase the number of nonzero entries.

B Algorithms

We present the various SONU variants to illustrate the evolution of its design. Alg. 2 shows the base SONU algorithm with fixed sketching matrices and without momentum. Alg. 3 uses evolving sketching matrices without momentum. Alg. 4 incorporates momentum with fixed sketching matrices, while Alg. 5 combines momentum with evolving sketching matrices. Alg. 6 further adds sketch initialization and progressive orthonormalization without error feedback.

C LoRA Realization of SONU

The sketching operations in SONU can be implemented with a LoRA-style parameterization without materializing or communicating the dense gradient. To make the distinction between the mathematical update and its implementation explicit, let $\Theta \in \mathbb{R}^{I\times J}$ denote the *effective* dense weight used in the forward pass and let W_{base} denote the stored base parameter. We parameterize

$$\Theta = W_{\mathrm{base}} + BA^\top, \qquad B \in \mathbb{R}^{I\times d}, \quad A \in \mathbb{R}^{J\times d}, \tag{15}$$

Algorithm 3 SONU with Evolving Sketching Matrices

Input: Data shards $\{\mathcal{D}_k\}_{k=1}^K$, initial weights W, sketching matrices P^L, P^R, learning rate η, switching period S, sketch dimension d, evolution rate α
Output: Updated W, P^L, P^R

for $t = 1, 2, \ldots, T$ **do**
  for each worker $k = 1, \ldots, K$ in parallel **do**
    Sample minibatch $\mathcal{B}_k \subset \mathcal{D}_k$
    Compute local gradient sketches S_k^L, S_k^R
  end for
  Master averages local sketches $S^L = \frac{1}{K}\sum_k S_k^L$, $S^R = \frac{1}{K}\sum_k S_k^R$
  Construct $\hat{G} \leftarrow S^L((P^L)^\top S^L)^\dagger (S^R)^\top$
  Compute orthonormal factors $U, V = \mathrm{Orth}(\hat{G})$
  if $\lfloor t/S \rfloor$ is even **then**
    Update left sketch $P^L \leftarrow P^L - \eta\alpha U$
    Keep P^R fixed
  else
    Update right sketch $P^R \leftarrow P^R - \eta\alpha V$
    Keep P^L fixed
  end if
  Update base weight $W \leftarrow W - \eta UV^\top$
end for

Algorithm 4 SONU with Momentum on Fixed Sketching Matrices

Input: Data shards $\{\mathcal{D}_k\}_{k=1}^K$, initial weights W, sketching matrices P^L, P^R, learning rate η, momentum β, sketch dimension d
Output: Updated W

Initialize projected momentum buffers $\bar{M}^L \leftarrow 0 \in \mathbb{R}^{I\times d}$ and $\bar{M}^R \leftarrow 0 \in \mathbb{R}^{J\times d}$
for $t = 1, 2, \ldots, T$ **do**
  for each worker $k = 1, \ldots, K$ in parallel **do**
    Sample minibatch $\mathcal{B}_k \subset \mathcal{D}_k$
    Compute local gradient sketches S_k^L, S_k^R
  end for
  Master averages local sketches $S^L = \frac{1}{K}\sum_k S_k^L$, $S^R = \frac{1}{K}\sum_k S_k^R$
  Update projected momentum buffers:
  $\bar{M}^L \leftarrow \beta\bar{M}^L + S^L, \qquad \bar{M}^R \leftarrow \beta\bar{M}^R + S^R$
  Construct $\hat{M} \leftarrow \bar{M}^L((P^L)^\top \bar{M}^L)^\dagger (\bar{M}^R)^\top$
  Compute orthonormal factors $U, V = \mathrm{Orth}(\hat{M})$
  Update $W \leftarrow W - \eta UV^\top$
end for

where B realizes the left sketching matrix P^L and A realizes the right sketching matrix P^R. Thus, the dense weight denoted by W in the main text corresponds to Θ in this appendix. We assume a unit LoRA scaling coefficient; any fixed coefficient can be absorbed into A, B, and the compensation formulas below.

Obtaining both gradient sketches in one backward pass. For worker k, let $G_k := \nabla_\Theta \mathcal{L}_k \in \mathbb{R}^{I\times J}$. Applying the chain rule to (15) gives

$$\nabla_B \mathcal{L}_k = G_k A, \qquad \nabla_A \mathcal{L}_k = G_k^\top B. \tag{16}$$

Consequently, after averaging across workers,

$$S^L = \frac{1}{K}\sum_{k=1}^K \nabla_B \mathcal{L}_k = GA, \qquad S^R = \frac{1}{K}\sum_{k=1}^K \nabla_A \mathcal{L}_k = G^\top B, \tag{17}$$

which are exactly the two sketches in (1) under the identification $P^L = B$ and $P^R = A$. In an implementation, W_{base} is frozen during automatic differentiation, while A and B require gradients.

Algorithm 5 SONU with Momentum on Evolving Sketching Matrices

Input: Data shards $\{\mathcal{D}_k\}_{k=1}^K$, initial weights W, sketching matrices P^L, P^R, learning rate η, momentum β, switching period S, sketch dimension d, evolution rate α
Output: Updated W, P^L, P^R

Initialize projected momentum buffers $\bar{M}^L \leftarrow 0 \in \mathbb{R}^{I\times d}$ and $\bar{M}^R \leftarrow 0 \in \mathbb{R}^{J\times d}$
Initialize reference sketches $P^L_{\text{ref}} \leftarrow P^L$ and $P^R_{\text{ref}} \leftarrow P^R$
for $t = 1, 2, \ldots, T$ **do**
  for each worker $k = 1, \ldots, K$ in parallel **do**
    Sample minibatch $\mathcal{B}_k \subset \mathcal{D}_k$
    Compute local gradient sketches S^L_k, S^R_k
  end for
  Master averages local sketches $S^L = \frac{1}{K}\sum_k S^L_k$, $S^R = \frac{1}{K}\sum_k S^R_k$
  Compute alignment maps $C^R \leftarrow ((P^R_{\text{ref}})^\top P^R_{\text{ref}})^\dagger (P^R_{\text{ref}})^\top P^R$ and $C^L \leftarrow ((P^L_{\text{ref}})^\top P^L_{\text{ref}})^\dagger (P^L_{\text{ref}})^\top P^L$
  Update aligned projected momentum buffers: $\bar{M}^L \leftarrow \beta \bar{M}^L C^R + S^L, \quad \bar{M}^R \leftarrow \beta \bar{M}^R C^L + S^R$
  Construct $\hat{M} \leftarrow \bar{M}^L((P^L)^\top \bar{M}^L)^\dagger (\bar{M}^R)^\top$
  Compute orthonormal factors $U, V = \mathrm{Orth}(\hat{M})$
  Refresh reference sketches $P^L_{\text{ref}} \leftarrow P^L$ and $P^R_{\text{ref}} \leftarrow P^R$
  if $\lfloor t/S \rfloor$ is even **then**
    Update left sketch $P^L \leftarrow P^L - \eta\alpha U$
    Keep P^R fixed
  else
    Update right sketch $P^R \leftarrow P^R - \eta\alpha V$
    Keep P^L fixed
  end if
  Update base weight $W \leftarrow W - \eta U V^\top$
end for

The linear layer can be evaluated in the usual fused LoRA form, so neither Θ nor its dense gradient needs to be materialized as an additional buffer. The factor gradients are used as SONU sketches rather than being passed to a conventional optimizer.

Initialization without changing the pretrained model. Unlike standard LoRA initialization, SONU requires both factors to be nonzero so that both sketches in (17) are informative from the first iteration. Let $\Theta_0 = U_0\Sigma_0 V_0^\top$ be the SVD of the pretrained dense weight. The initialization in (13) is realized by

$$B_0 = \gamma U_0[:, :d], \qquad A_0 = \gamma V_0[:, :d]. \tag{18}$$

Because $B_0 A_0^\top$ is nonzero, directly inserting these factors would perturb the pretrained model. We therefore compensate the stored base weight once at initialization:

$$W_{\text{base},0} = \Theta_0 - B_0 A_0^\top. \tag{19}$$

It follows that $W_{\text{base},0} + B_0 A_0^\top = \Theta_0$, so the forward function is unchanged while the desired sketching subspaces are available immediately.

General base-weight compensation rule. Suppose SONU produces paired orthonormal factors U and V and intends to apply the effective dense update

$$\Theta^+ = \Theta - \eta U V^\top. \tag{20}$$

After choosing updated sketch factors B^+ and A^+, the stored base weight must be set to

$$W^+_{\text{base}} = W_{\text{base}} - \eta U V^\top + B A^\top - B^+ (A^+)^\top. \tag{21}$$

Indeed, substituting (21) into the merged parameterization yields

$$W^+_{\text{base}} + B^+ (A^+)^\top = W_{\text{base}} + B A^\top - \eta U V^\top = \Theta - \eta U V^\top. \tag{22}$$

Thus, the factor evolution changes only the sketching coordinates; the effective model follows exactly the dense SONU update. When A and B are fixed, the correction term vanishes and the implementation reduces to $W^+_{\text{base}} = W_{\text{base}} - \eta U V^\top$.

Algorithm 6 SONU with Sketch Initialization and Progressive Orthonormalization

Input: Data shards $\{\mathcal{D}_k\}_{k=1}^K$, initial weight W, learning rate η, momentum β, switching period S, sketch dimension d, evolution rate α, initialization scale γ
Output: Updated W, P^L, P^R

Initialize sketching matrices P^L, P^R with Eq. (13)
Initialize projected momentum buffers $\bar{M}^L \leftarrow 0 \in \mathbb{R}^{I\times d}$ and $\bar{M}^R \leftarrow 0 \in \mathbb{R}^{J\times d}$
Initialize reference sketches $P^L_{\text{ref}} \leftarrow P^L$ and $P^R_{\text{ref}} \leftarrow P^R$
for $t = 1, 2, \ldots, T$ **do**
  for each worker $k = 1, \ldots, K$ in parallel **do**
    Sample minibatch $\mathcal{B}_k \subset \mathcal{D}_k$
    Compute local gradient sketches S^L_k, S^R_k
  end for
  Master averages local sketches $S^L = \frac{1}{K}\sum_k S^L_k$, $S^R = \frac{1}{K}\sum_k S^R_k$
  Compute alignment maps:
  $C^R \leftarrow ((P^R_{\text{ref}})^\top P^R_{\text{ref}})^\dagger (P^R_{\text{ref}})^\top P^R, \qquad C^L \leftarrow ((P^L_{\text{ref}})^\top P^L_{\text{ref}})^\dagger (P^L_{\text{ref}})^\top P^L$
  Update aligned projected momentum buffers:
  $\bar{M}^L \leftarrow \beta \bar{M}^L C^R + S^L, \qquad \bar{M}^R \leftarrow \beta \bar{M}^R C^L + S^R$
  Construct $\hat{M} \leftarrow \bar{M}^L((P^L)^\top \bar{M}^L)^\dagger (\bar{M}^R)^\top$
  Compute orthonormal factors $U, V = \mathrm{Orth}(\hat{M})$
  Refresh reference sketches $P^L_{\text{ref}} \leftarrow P^L$ and $P^R_{\text{ref}} \leftarrow P^R$
  if $\lfloor t/S \rfloor$ is even **then**
    Evolve the left sketch $\widetilde{P}^L \leftarrow P^L - \eta\alpha U$
    Progressively orthonormalize $P^L \leftarrow \gamma\,\mathrm{Orth}(\widetilde{P}^L)$
    Keep P^R fixed
  else
    Evolve the right sketch $\widetilde{P}^R \leftarrow P^R - \eta\alpha V$
    Progressively orthonormalize $P^R \leftarrow \gamma\,\mathrm{Orth}(\widetilde{P}^R)$
    Keep P^L fixed
  end if
  Update base weight $W \leftarrow W - \eta UV^\top$
end for

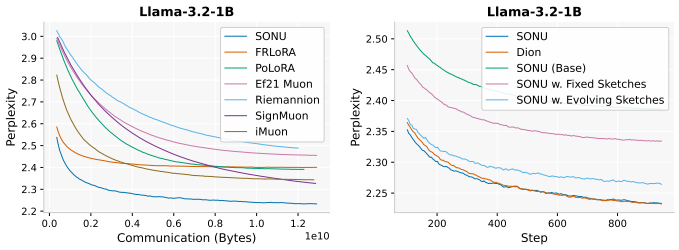
Algorithm 7 Dion

Input: Data shards $\{\mathcal{D}_k\}_{k=1}^K$, initial weights W, momentum M, semi-orthonormal matrix V, learning rate η, momentum parameter β, rank R
Output: Updated W, M, V

for $t = 1, 2, \ldots, T$ **do**
  for each worker $k = 1, \ldots, K$ in parallel **do**
    Sample minibatch $\mathcal{B}_k \subset \mathcal{D}_k$
    Compute local gradient $G_k \in \mathbb{R}^{I\times J}$
  end for
  Average gradient $G \leftarrow \frac{1}{K}\sum_{k=1}^K G_k$
  Update momentum buffer $M \leftarrow M + G$
  Compute compressed projection $P \leftarrow MV$
  Orthonormalize projected matrix $U \leftarrow \mathrm{Orth}(P)$
  Project back to the column space $W_{\text{proj}} \leftarrow M^\top U$
  Apply error-feedback update $M \leftarrow M - \beta U W_{\text{proj}}^\top$
  Normalize columns $V \leftarrow \mathrm{ColNorm}(W_{\text{proj}})$
  Form orthonormalized update $O \leftarrow UV^\top$
  Update weights $W \leftarrow W - \eta O$
end for

Alternating evolution of the LoRA realization. When evolving the left sketch, we set $B^+ = B - \alpha\eta U$ and $A^+ = A$. Applying (21) gives

$$B^+ = B - \alpha\eta U, \qquad A^+ = A, \qquad W^+_{\text{base}} = W_{\text{base}} - \eta U(V - \alpha A)^\top. \tag{23}$$



(a) (b)

Figure 5: Perplexity trajectory on CoT Training Task. SONU achieves the lowest perplexity among baselines and closely tracks Dion's trajectory. Ablation study in 5b demonstrates the gains from each component of SONU's design.

Symmetrically, evolving the right sketch with $A^+ = A - \alpha\eta V$ and $B^+ = B$ gives

$$B^+ = B, \qquad A^+ = A - \alpha\eta V, \qquad W^+_{\text{base}} = W_{\text{base}} - \eta(U - \alpha B)V^\top. \tag{24}$$

Equations (23) and (24) are the LoRA realization of the sketch updates in (4) and (5), respectively. The compensation terms $\alpha\eta UA^\top$ and $\alpha\eta BV^\top$ cancel the change in the low-rank product, ensuring that updating a sketching factor does not introduce an unintended model update. The two branches are alternated every S rounds as in Algorithm 3.

Progressive orthonormalization. Factor evolution can gradually degrade the geometry of the sketching matrices. For a left-sketch update, define

$$\widetilde{B} = B - \alpha\eta U, \qquad Q_B = \text{Orth}(\widetilde{B}), \qquad B^+ = \gamma Q_B, \qquad A^+ = A. \tag{25}$$

The corresponding compensated base update is

$$W^+_{\text{base}} = W_{\text{base}} - \eta UV^\top + (B - \gamma Q_B)A^\top. \tag{26}$$

For the symmetric right-sketch update,

$$\widetilde{A} = A - \alpha\eta V, \qquad Q_A = \text{Orth}(\widetilde{A}), \quad A^+ = \gamma Q_A, \quad B^+ = B, \tag{27}$$

$$W^+_{\text{base}} = W_{\text{base}} - \eta UV^\top + B(A - \gamma Q_A)^\top. \tag{28}$$

Both branches satisfy (20) exactly. The additional compensation has rank at most d, so the complete base update has rank at most $2d$. In practice, the QR factorization used to obtain Q_A or Q_B is local and requires no additional communication. These updates implement the progressive orthonormalization in (14) while preserving the merged model.

Momentum alignment in factor notation. The projected momentum buffers use the synchronized factor gradients directly: $\bar{M}^L$ accumulates $\nabla_B\mathcal{L}$, while $\bar{M}^R$ accumulates $\nabla_A\mathcal{L}$. When the factors evolve, the alignment maps in Algorithm 5 become

$$C^R = (A^\top_{\text{ref}}A_{\text{ref}})^\dagger A^\top_{\text{ref}}A, \qquad C^L = (B^\top_{\text{ref}}B_{\text{ref}})^\dagger B^\top_{\text{ref}}B, \tag{29}$$

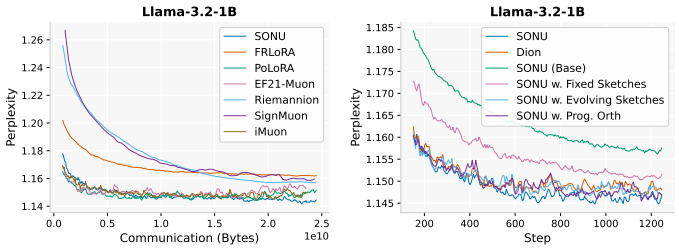
$$\bar{M}^L \leftarrow \beta\bar{M}^L C^R + \nabla_B\mathcal{L}, \qquad \bar{M}^R \leftarrow \beta\bar{M}^R C^L + \nabla_A\mathcal{L}, \tag{30}$$

where $\nabla_B L := (1/K)\sum_k \nabla_B L_k = S^L$. Any residual accumulation or error-feedback mechanism is applied on the server to the projected momentum state constructed from these synchronized sketches; it is independent of the LoRA realization and base-weight compensation described above.

D Additional Experimental Results

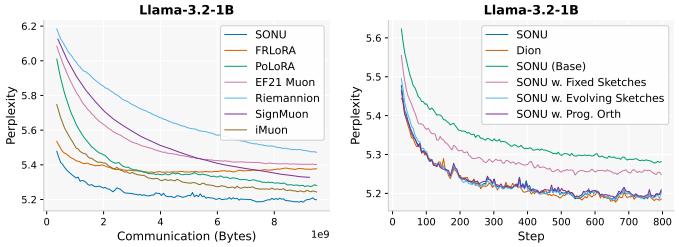
D.1 Discussions on Baselines

We discuss additional important facts of the baseline methods.



(a) (b)

Figure 6: Perplexity trajectory on Agentic Function Calling Task. SONU achieves the lowest or near-lowest perplexity among baselines and closely tracks Dion's trajectory. Ablation study in 6b demonstrates the gains from each component of SONU's design.



(a) (b)

Figure 7: Perplexity trajectory on Instruction Following Task. SONU achieves the lowest perplexity among baselines and closely tracks Dion's trajectory. Ablation study in 7b demonstrates the gains from each component of SONU's design.

Dion. We set Dion's rank at 16 to match the downlink communication overhead with other baselines. We acknowledge that Dion supports communication-efficient synchronization by exchanging projected factors. However, in our master–worker setting, Dion with master-held momentum requires dense gradient uplink. Dion's compressed synchronization (Ahn et al., 2025, §3.3) avoids this, but it keeps a dense, decoupled $I \times J$ momentum for each optimized weight matrix on every worker, incurring $O(IJ)$ memory despite the reduced communication volume. SONU addresses this remaining worker-side cost: workers compute and communicate two-sided gradient sketches through LoRA factor gradients, while momentum accumulation and residual maintenance are performed on the server. Consequently, SONU avoids materializing dense gradients or maintaining dense momentum on workers, and the sketch gradient is automatically derived from back propagation, excluding model weights and activations. This distinction is particularly relevant when both network bandwidth and worker memory are constrained. Accordingly, the main-text comparison uses a common worker-memory budget, under which we evaluate Dion with standard dense-gradient synchronization as in Alg. 7. Since Dion retains the complete dense momentum in either configuration, it serves as a full-information reference, and the comparison contrasts SONU's sketched information with Dion's full momentum information.

EF21 Muon. EF21-Muon reduces communication through error feedback while retaining full-dimensional optimizer state on workers. In its standard formulation, each worker maintains a local momentum buffer and a gradient estimator used to compute compressed corrections. Although the transmitted corrections may be sparse, the accumulated estimator can become dense, so these buffers require $O(IJ)$ storage for each optimized $I \times J$ weight matrix. In this work, we adopt the standard Top-K for sparsification with compression rate identical to other baselines, so the communication overhead per step also stays the same with other baselines.

SignMuon. SignMuon communicates the entrywise signs of locally computed orthonormalized directions. Under the ideal one-bit encoding of a BF16 training, this yields a fixed communication ratio of $1/16 = 6.25\%$, exceeding the 1.30% and 2.03% ratios used for the other compressed baselines in our experiments. SignMuon therefore receives a larger per-iteration communication

Table 1: General Hyperparameter settings.

Hyperparameter	Value
Learning Rate	$\{5 \times 10^{-5}, 1 \times 10^{-4}, 5 \times 10^{-4}, 1 \times 10^{-3}, 5 \times 10^{-3}, 1 \times 10^{-2}, 5 \times 10^{-2}\}$
Momentum	0.9
Batch Size Per Worker	2
Scheduler	Cosine
Scheduler Min LR	1×10^{-5}

Table 2: SONU Hyperparameter settings.

Hyperparameter	Value
Update multiplier α	$\{10, 25, 50\}$
Switching interval S	$\{10, 50\}$
Initialization scale γ	$\{0.1, 1.0, 5.0\}$

budget under this accounting. To account for these differences, we report validation performance against cumulative communication volume, rather than training iterations.

FRLoRA. FRLoRA computes updates through LoRA factor optimization and merges a global residual term $B_t^G (A_t^G)^\top - B_t A_t^\top$ into the base weight at each communication round, where B_t^G and A_t^G denote the globally aggregated factors, and B_t and A_t denote their shared initialization for that round. Accumulating these residual updates allows the cumulative weight change to exceed the adapter rank. We also acknowledge ReLoRA (Lialin et al., 2024), which similarly enables higher-rank weight changes by periodically merging low-rank updates into the base weights and reinitializing the adapters, together with partial optimizer-state resets and learning-rate re-warmup. We include FRLoRA as a representative baseline for accumulating LoRA-based updates because its explicit factor aggregation and per-round residual merging naturally fit our master–worker communication protocol.

We apply a single rule to all baselines: whenever a method admits a configuration without persistent dense optimizer state on workers, we evaluate that configuration, which makes the baseline comparable with SONU. Dion admits such a configuration, so we evaluate Dion with server-held momentum (Alg. 7). EF21-Muon and SignMuon, in contrast, have no such configuration: EF21-Muon's error feedback requires each worker to maintain a local momentum buffer and gradient estimator, and SignMuon requires each worker to form a local orthonormalized direction from its own momentum. We therefore evaluate them in their standard form.

D.2 Experimental Settings.

In this paper, we adopt the standard master-worker setting. This setting matches bandwidth-limited deployments where workers connect to a coordinator over slow links. The experiments are performed on NVIDIA-A100 GPUs. We split the dataset with a ratio of 95:5 as the training/validation split. The pseudo-inverse is computed with a relative tolerance of 10^{-6}. The hyperparameter budgets shared by all methods are presented in Table 1. The hyperparameter budgets specific to SONU are presented in Table 2, and the selected hyperparameters used in the experiments are reported in Table 3. For all baselines that need Newton-Schulz iterations to do orthonormalization, we use the standard NS5 iterations. The learning rate selected for each baseline is presented in Table 4.

D.3 Ablation Studies on Llama-3.2-1B

We present the ablation studies on Llama-3.2-1B under the same configuration in the main paper. The experimental results are plotted in Fig. 5, 6, 7. The observation on step-by-step improvement of our algorithmic design is consistent with the Llama-3.2-3B experiments.

Table 3: SONU hyperparameter settings across tasks and model scales.

Hyperparameter	CoT Training		Agentic Function Calling		Instruction Following	
	1B	3B	1B	3B	1B	3B
Learning rate η	5e-3	5e-3	1e-3	1e-3	1e-3	1e-3
Update multiplier α	25	10	50	50	50	10
Switching interval S	50	50	50	50	50	50
Initialization scale γ	1.0	1.0	1.0	1.0	0.1	1.0

Table 4: Selected Learning Rates for Baselines.

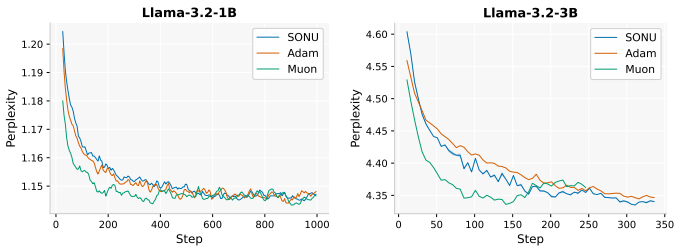
Method	CoT Training		Agentic Function Calling		Instruction Following	
	1B	3B	1B	3B	1B	3B
FRLoRA	1e-4	1e-4	1e-4	1e-4	1e-4	1e-3
PoLoRA	5e-3	5e-3	1e-2	1e-2	5e-3	5e-3
EF21 Muon	5e-4	5e-4	1e-3	1e-3	1e-4	1e-3
Riemannion	1e-4	1e-4	1e-4	5e-4	1e-4	1e-3
Sign Muon	5e-4	5e-4	5e-4	1e-3	5e-4	5e-4
iMuon	1e-3	1e-3	5e-3	1e-2	1e-3	5e-3

D.4 Comparing SONU with Full Adam and Muon

To evaluate the gap between SONU and uncompressed reference performance, we compare SONU against full Adam and Muon, both of which optimize in the full ambient dimension of the target modules. We evaluate on two tasks: Agentic Function Calling using Llama-3.2-1B and Instruction Following using Llama-3.2-3B. As shown in Figures 8a and 8b, SONU closely approaches the performance of both ambient-space baselines, achieving highly comparable final perplexities across both settings. In terms of step-wise convergence, Muon converges fastest (approximately $2\times$ faster than the others), while Adam and SONU exhibit nearly identical convergence trajectories. Crucially, because these figures align methods by optimization steps and both Adam and Muon must transmit full ambient gradient matrices, these results demonstrate that SONU achieves close performance while drastically reducing per-iteration communication overhead.

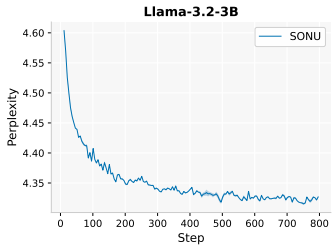
D.5 On the Sensitivity of SONU

To assess sensitivity to training randomness, we conduct three independent runs of SONU on the instruction-following task using Llama-3.2-3B. Figure 9a presents the validation perplexity trajectories across these runs. We compute the standard deviation across runs at each evaluation step. Over the training trajectory, the maximum standard deviation is 4.9×10^{-3}, and the average across



(a) (b)

Figure 8: Perplexity trajectory on Agentic Function Calling and Instruction Following Task. SONU achieves comparable perplexity with Adam and Muon, while Muon converges faster. Trajectories for Muon are truncated for better demonstration as later iterations exhibits instability.



(a)

Figure 9: Perplexity trajectory on Instruction Following Task, averaged over three independent runs. Shaded area represents the range with 1 standard deviation.

evaluation steps is 1.3×10^{-3}, which is relatively small compared to the absolute perplexity values. The maximum occurs around step 460, midway through training. This suggests that the increased variability during intermediate training is transient, with the runs attaining similar final validation perplexities. These results indicate low run-to-run variability in the evaluated setting, providing preliminary evidence that SONU's validation performance is robust to training randomness.

E Broader Impacts

By showing that the empirical performance of training with dense gradients can be achieved using only low-rank operations, SONU can democratize advanced LLM development. It empowers researchers with constrained computing clusters to train and study state-of-the-art foundational models at a scale and quality previously reserved for massive, highly-capable compute clusters. No prominent negative impacts can be foreseen other than general concerns of machine learning techniques.