GIFT: GEOMETRY-INFORMED LOW-PRECISION GRADIENT COMMUNICATION FOR LLM PRETRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Gradient communication is a major scaling bottleneck in large language model (LLM) pretraining. Low-precision formats such as FP8 can substantially reduce communication cost, but often at the expense of gradient fidelity. Existing approaches either quantize gradients directly in their original coordinates or apply data-independent transformations to suppress outliers. Our study shows that both approaches overlook a fundamental source of quantization distortion: LLM gradients are highly anisotropic, with geometry that varies substantially across layers, making low-precision error strongly direction dependent. This observation suggests that gradient fidelity depends not only on the quantizer itself, but also on the coordinates in which quantization is performed. Motivated by this insight, we design GIFT, a geometry-informed gradient communication method that transforms gradients into more isotropic coordinates before low-precision quantization. To make this transformation practical, GIFT uses a one-sided, low-rank approximation of the geometry and applies it only to layers that are most sensitive to quantization errors, while leaving the optimizer, training recipe, communication collective, and low-precision format unchanged. Across Llama-style models with approximately 600M and 2B parameters, GIFT achieves downstream task performance comparable to Muon training with FP32 gradients. On 64 NVIDIA GH200 Superchips, GIFT reduces end-to-end training step time relative to FP32 gradient communication by 10.87% for Llama-600M and 17.05% for Llama-2B.

1 INTRODUCTION

As large language model (LLM) pretraining scales across increasingly large GPU clusters, communication can consume a substantial fraction of end-to-end training time Narayanan et al. (2021); Song et al. (2023). For GPT-8.3B pretraining on 128 NVIDIA A100 GPUs, prior work reports that communication accounts for approximately 40% of the overall training time Song et al. (2023). In data-parallel training, an important component of this cost is gradient synchronization, where gradients must be communicated across participating GPUs before the model update. Reducing the amount of data exchanged during this synchronization is therefore an important direction for improving distributed training efficiency Alistarh et al. (2017); Lin et al. (2018); Vogels et al. (2019). In this work, we focus specifically on the data-parallel gradient synchronization path.

Gradient communication can be compressed through several approaches, including sparsification, low-rank approximation, and reduced-precision representation. We focus on reduced precision because it preserves dense collective communication and can directly benefit from emerging hardware and communication-library support. In particular, native FP8 support was introduced in NCCL 2.24 (January 2025) Williams et al. (2025), making FP8 gradient synchronization increasingly practical at scale. Reducing gradients from FP32 to FP8 cuts the communicated payload by 75%, but this reduction is useful only if the resulting low-precision gradients retain the information needed for training.

Existing low-precision methods improve this tradeoff by making better use of the numerical range. FP8-LM Peng et al. (2023) relies on scaling for FP8 training, COAT Xi et al. (2024) combines dynamic-range adaptation with mixed-granularity quantization, and SDP4Bit Jia et al. (2024) uses Hadamard-based smoothing for low-bit communication. These techniques mitigate overflow, outliers, or scale imbalance, but they do not adapt the quantization coordinates to the local geometry of

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.

the gradients. This raises a different question: even with a well-chosen scale, how much information is lost when quantization is performed in coordinates that are poorly matched to the local geometry of the gradients?

Our analysis shows that this issue is pronounced in LLM training. Within a layer, gradient variation can differ sharply across directions, while the dominant geometry also changes substantially from layer to layer. This directional concentration is consistent with prior observations of structured and anisotropic stochastic-gradient spectra Xie et al. (2023); Huang et al. (2026). Under standard Euclidean scaling, FP8 quantization is performed directly in the original coordinates without accounting for this directional imbalance. Consequently, quantization distortion can vary substantially across directions even when the overall tensor range is well covered. The resulting error is therefore not only a consequence of insufficient bit width or an imperfect scale, it also depends on the coordinates in which quantization is performed. We compute the same geometry statistics over all 64 MLP projections of Llama-600M. Figure 1 shows the projection with the largest relative increase in NERank after the geometry-informed transformation. Complete layer-wise results are provided in Appendix B.3. In this example, projected gradients are strongly anisotropic in the original Euclidean coordinates, whereas the geometry-informed transformation distributes their variation much more evenly across directions.

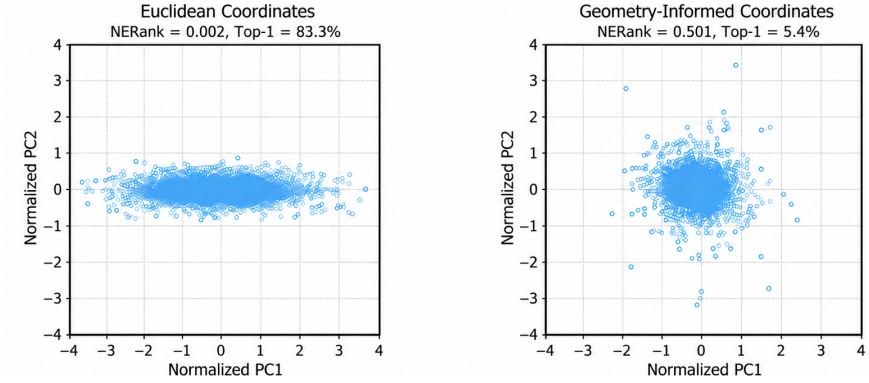


Figure 1: Projected gradient geometry of the same MLP FC2 layer (Layer 25, Llama-600M) in Euclidean coordinates (left) and after the geometry-informed transformation (right). Both panels use the same gradient samples and normalized PCA axes with the same range. NERank denotes the normalized effective rank Roy & Vetterli (2007), while Top-1 denotes the fraction of spectral energy in the leading singular direction; higher NERank and lower Top-1 indicate less directionally concentrated, more isotropic geometry.

This motivates a central principle of our approach: the communication representation itself should be treated as part of the quantization design. Specifically, we use a K-FAC approximation Amari (2016; 1998); Martens & Grosse (2015) to characterize local gradient geometry, but only along the communication path: the optimizer and the resulting Muon update Jordan et al. (2024) remain unchanged. GIFT is compatible with other parallel training paradigms, such as pipeline-, tensor-, context-, and expert-parallel, because it only modifies the gradient communication path.

Applying the full geometry transformation to every layer improves gradient fidelity but introduces additional computation to offset the communication benefit. To make the approach practical, GIFT therefore uses a one-sided, low-rank approximation of the geometry and applies it only to layers responsible for most of the observed quantization error, while the remaining layers use standard Euclidean FP8 communication. This design efficiently allocates the additional computation where changing coordinates yields the largest fidelity benefit.

We implement GIFT in Megatron-LM Narayanan et al. (2021) and evaluate it on Llama-style models Touvron et al. (2023) with approximately 600M and 2B parameters trained on OpenWebText Gokaslan et al. (2019). We compare against Euclidean FP8 scaling at multiple granularities, error-feedback variants, and representative low-precision communication baselines under matched Muon training. Across these settings, GIFT reduces relative gradient reconstruction error by 52.7% on average compared with Euclidean FP8 and provides stronger downstream-task preservation. These fidelity gains are achieved while retaining substantial end-to-end speedups over FP32 gradient communication. GIFT matches or exceeds Muon FP32 on 8/14 and 7/14 downstream tasks

for Llama-600M and Llama-2B, respectively. On 64 NVIDIA GH200 Superchips, GIFT reduces end-to-end training step time relative to FP32 gradient communication by 10.87% for Llama-600M and 17.05% for Llama-2B.

This paper makes three main contributions:

A geometric explanation for low-precision gradient distortion. We identify strong directional imbalance in LLM gradients as a source of FP8 communication error and mathematically show why quantization distortion depends on the coordinates in which gradients are represented.

A practical geometry-informed communication design. We introduce **GIFT**, which constructs communication coordinates using a lightweight one-sided, low-rank approximation of gradient geometry and selectively applies the transformation to layers with the largest quantization errors.

An improved fidelity–efficiency tradeoff. Across multiple FP8 communication baselines and two model scales, GIFT substantially improves gradient fidelity and downstream preservation while retaining meaningful end-to-end speedups over FP32 gradient communication.

2 Related Work

This section reviews prior work in three closely related areas: gradient communication compression, low-precision LLM pretraining, and geometry-aware optimization.

2.1 Gradient Communication Compression

A long line of work reduces distributed-training overhead by compressing communicated gradients through quantization, sparsification, low-rank approximation, or optimizer-specific designs Peng et al. (2026); Dettmers (2016); Faghri et al. (2020); Alistarh et al. (2017); Lin et al. (2018); Vogels et al. (2019); Tang et al. (2021); Wang et al. (2023); Xie et al. (2024); Song et al. (2023); Jia et al. (2024); Li et al. (2024). QSGD and TernGrad reduce communication through gradient quantization, DGC exploits gradient sparsity, PowerSGD communicates low-rank gradient approximations, and 1-bit Adam couples aggressive compression with Adam-style optimization Zhao et al. (2025); Wen et al. (2017); Seide et al. (2014); Dettmers (2016); Alistarh et al. (2017); Lin et al. (2018); Vogels et al. (2019); Tang et al. (2021). Recent systems further target low-bit LLM training in sharded or hierarchical communication settings Markov et al. (2023); Wang et al. (2023); Xie et al. (2024); Song et al. (2023); Jia et al. (2024). LoCo, for example, improves low-bit communication through error compensation, while SDP4Bit combines low-bit gradient communication with smoothing and hierarchical communication Xie et al. (2024); Jia et al. (2024).

In contrast, GIFT does not sparsify gradients, communicate a low-rank representation, or introduce a new collective. Instead, it uses a low-rank approximation only to construct the geometry that determines where dense gradients are quantized and communicated in FP8. Thus, GIFT treats the communication coordinate system as a design dimension for low-precision gradient compression.

2.2 Low-Precision LLM Pretraining

Low-precision training has progressed from mixed-precision computation to FP8-based LLM pretraining Lin et al. (2026); Micikevicius et al. (2018; 2022); Perez et al. (2023); Peng et al. (2023); Xi et al. (2024); Wang et al. (2018). These methods typically use scaling and dynamic-range control to map tensors to the representable range of low-precision formats Micikevicius et al. (2022); Peng et al. (2023); Xi et al. (2024). FP8-LM extends FP8 to multiple training tensors, including gradients and optimizer states, while COAT further develops dynamic-range adaptation and mixed-granularity quantization for activations and optimizer states Peng et al. (2023); Xi et al. (2024). SDP4Bit instead targets distributed communication, compressing both weight and gradient communication with low-bit quantization and Hadamard-based smoothing Jia et al. (2024).

These approaches primarily improve how tensors are scaled, smoothed, or represented within a fixed communication coordinate system. In contrast, GIFT changes the coordinates in which gradient quantization is performed using local gradient geometry.

2.3 Natural-Gradient and Curvature-Aware Optimization

Natural-gradient and curvature-aware optimization methods incorporate local geometric or curvature information into parameter updates Amari (1998); Martens & Grosse (2015); Eschenhagen et al. (2023); Ba et al. (2017); Gupta et al. (2018). Natural-gradient methods define update directions using the Fisher information metric, while K-FAC provides a tractable approximation by representing layer-wise Fisher blocks with Kronecker-factored structure Amari (1998); Martens & Grosse (2015). These methods use such information primarily to improve optimization by modifying the parameter-update rule.

GIFT draws on the same geometric framework but uses it for a different purpose. We adopt K-FAC because its Kronecker-factored structure provides a tractable layer-wise approximation to the Fisher geometry while exposing separate input- and output-side second-order structure. This factorization allows GIFT to construct temporary geometry-aware coordinates for low-precision gradient communication without changing the optimizer.

3 GIFT: Geometry-Informed Communication Design

The design of GIFT begins with a simple question: can FP8 distortion be reduced by changing the coordinates in which gradients are communicated?

3.1 Layer-Wise Anisotropy and Communication Coordinates

Observation 3.1 (Layer-wise anisotropic distortion)**.** LLM gradients exhibit strongly anisotropic geometry, as illustrated in Fig. 1 for an MLP layer, and this geometry varies across layers. Consequently, direct low-precision quantization can incur different distortion across directions and layers. This motivates adapting the communication representation to each layer's local geometry.

Observation 3.2 (Coordinate-dependent quantization)**.** For the weight gradient $W_g^{(l)}$ of layer l, let $\mathcal{T}_l$ be an invertible layer-specific transformation and let $\mathcal{Q}_{\mathrm{FP8}}$ denote FP8 quantization followed by dequantization. Quantizing in the transformed coordinates gives

$$\widehat{W}_g^{(l)} = \mathcal{T}_l^{-1}\left(\mathcal{Q}_{\mathrm{FP8}}\left(\mathcal{T}_l(W_g^{(l)})\right)\right). \tag{1}$$

This generally differs from direct quantization of $W_g^{(l)}$; Layer-wise Euclidean FP8 is the special case $\mathcal{T}_l = I$. The remaining question is how to choose $\mathcal{T}_l$ from the local geometry.

3.2 Geometry-Normalized Communication Coordinates

We use the Fisher metric to characterize local sensitivity across parameter directions. For a perturbation ΔW of layer weight W, let F_W denote its layer-wise Fisher block, with local metric $\|\Delta W\|_{F_W}^2 = \mathrm{vec}(\Delta W)^\top F_W \,\mathrm{vec}(\Delta W)$. The full Fisher can be approximated layer-wise, and K-FAC further factorizes each linear-layer block into input-activation and pre-activation-gradient statistics (Appendix A.1–A.2) Amari (2016; 1998); Martens & Grosse (2015).

Assumption 3.3 (Layer-wise K-FAC geometry)**.** For input activation $a \in \mathbb{R}^{d_{\mathrm{in}}}$ and pre-activation gradient $\delta \in \mathbb{R}^{d_{\mathrm{out}}}$, we use

$$F_W \approx A \otimes G, \qquad A = \mathbb{E}[aa^\top], \qquad G = \mathbb{E}[\delta\delta^\top], \tag{2}$$

where A and G are the input- and output-side covariance factors. With damping so that $A, G \succ 0$, let $A = L_A L_A^\top$ and $G = L_G L_G^\top$.

Lemma 3.4 (Metric normalization)**.** The K-FAC metric is approximately Euclidean under

$$Z = L_G^\top \Delta W L_A, \qquad \|\Delta W\|_{F_W}^2 \approx \|Z\|_F^2, \tag{3}$$

where Z is the transformed parameter perturbation and $\|\cdot\|_F$ is the Frobenius norm (Appendix A.3). Gradient communication requires the corresponding coordinates for the gradient itself.

Proposition 3.5 (Dual gradient coordinates)**.** Let $W_g = \nabla_W \mathcal{L}$ denote the gradient of the training loss. Preserving its first-order pairing with parameter perturbations uniquely gives

$$\widetilde{W}_g = \mathcal{T}_g(W_g) = L_G^{-1} W_g L_A^{-\top}, \tag{4}$$

where $\widetilde{W}_g$ is the gradient in the geometry-normalized coordinates (Appendix A.4). Thus, the inverse factors in Eq. 4 follow from the dual geometry rather than being an additional heuristic, fixing the transformation left open in Observation 3.2.

3.3 Geometry-Informed FP8 Communication

Definition 3.6 (Full geometry-informed gradient communication)**.** Using the dual gradient coordinates in Eq. 4, the full two-sided communication path is

$$W_g \xrightarrow{\mathcal{T}_g} \widetilde{W}_g \xrightarrow{\mathrm{Comm_{FP8}}} \widehat{\widetilde{W}}_g \xrightarrow{\mathcal{T}_g^{-1}} \widehat{W}_g, \qquad \mathcal{T}_g^{-1}(X) = L_G X L_A^\top. \tag{5}$$

Here, $\mathrm{Comm_{FP8}}(\cdot)$ denotes the all-reduce communication of gradients in FP8, and $\widehat{W}_g$ is the reconstructed synchronized gradient in the original parameter coordinates.

4 Implementation

Having established the full geometry-informed communication rule in §3, we now turn to its practical realization. As summarized in Fig. 2, the implementation addresses three main components: efficient approximation of the geometry transformation, selective application and maintenance of the layer-wise geometry, and geometry-aware FP8 gradient communication. Algorithm 1 summarizes the complete procedure. We implement GIFT in Megatron-LM Narayanan et al. (2021) by integrating this procedure into data-parallel gradient synchronization.

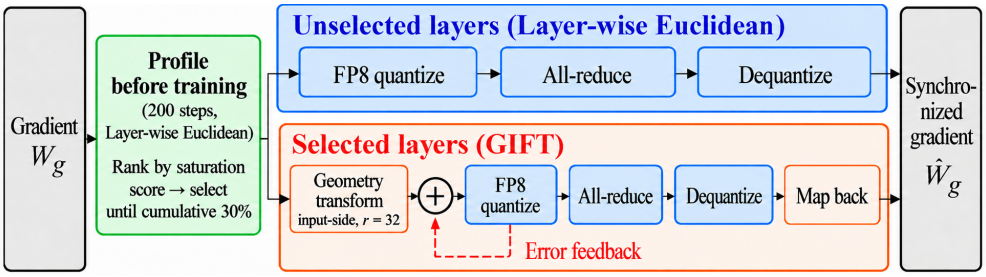


Figure 2: Overview of GIFT. Geometry-aware FP8 communication is selectively applied to sensitive layers, while the remaining layers retain standard Layer-wise Euclidean FP8 communication.

4.1 Efficient Geometry Approximation

The full two-sided communication path in Eq. 5 improves FP8 reconstruction fidelity, but directly applying it introduces additional computation that offsets part of the communication savings. We therefore make the transform practical in two stages: retaining only the input-side geometry and using a low-rank approximation with a diagonal residual correction.

Input-side geometry. Our reconstruction study in §5.2 shows that both one-sided factors improve FP8 reconstruction fidelity, while the full two-sided transformation provides the strongest reconstruction accuracy. Among the one-sided alternatives, the input-side factor provides the larger improvement (Appendix B.1). We therefore retain only the input-side geometry (Appendix A.5):

$$\widetilde{W}_g = W_g L_A^{-\top}, \qquad \widehat{W}_g = \widehat{\widetilde{W}}_g L_A^\top. \tag{6}$$

This removes the output-side factor from both the pre-communication transform and the post-communication reconstruction, while retaining most of the reconstruction benefit of the full two-sided geometry.

Low-rank geometry. To obtain a structured and stable approximation of the input-side geometry, GIFT retains the dominant rank-r spectral structure of A,

$$A \approx U_r \Lambda_r U_r^\top, \qquad r \ll d_{\mathrm{in}}, \tag{7}$$

where U_r contains the retained directions and Λ_r their corresponding spectral values. To obtain a stable and invertible communication transform, we use

$$\widetilde{A}_r = U_r \Lambda_r U_r^\top + D_r = L_{A,r} L_{A,r}^\top, \tag{8}$$

where $D_r \succ 0$ is a diagonally stabilized residual term that preserves per-coordinate information outside the rank-r subspace and ensures stability. We use the lower-triangular Cholesky factor $L_{A,r}$ of $\widetilde{A}_r$. The communication transform is $\widetilde{W}_g = W_g L_{A,r}^{-\top}$, with reconstruction $\widehat{W}_g = \widehat{\widetilde{W}}_g L_{A,r}^{\top}$. We use rank $r = 32$ in the main experiments. As shown in §5.2, this approximation retains most of the reconstruction benefit of the dense input-side transformation while using a simplified geometry representation. See Appendix B.2 for approximation comparisons and Appendix C.1 for implementation details.

4.2 Layer Selection and Geometry Maintenance

Even with the low-rank approximation, applying the geometry-aware path uniformly across layers introduces unnecessary overhead because quantization sensitivity is highly nonuniform. GIFT therefore applies the transformation only to layers where low-precision distortion is most pronounced.

Automatic layer selection. Before formal pretraining, GIFT performs a separate 200-step Layer-wise Euclidean FP8 profiling run over candidate MLP layers. For each candidate layer, we accumulate both the number of quantized gradient values that reach the positive or negative FP8 representable boundary and its quantization error over the profiling run. The cumulative boundary-hit count is used to rank layers by their sensitivity to FP8 saturation. Boundary hits indicate clipping at the FP8 representable limits, which can introduce substantially larger distortion than ordinary in-range rounding Micikevicius et al. (2022); Peng et al. (2023).

Layers are ranked in descending order of this count, and GIFT selects the smallest prefix whose cumulative quantization error accounts for at least 30% of the total quantization error across all candidate layers. The selected set $\mathcal{S}$ is then fixed throughout a separate pretraining run initialized from a fresh model. Layers outside $\mathcal{S}$ use standard Layer-wise Euclidean FP8 communication. The formal definitions and additional profiling details are provided in Appendix C.2.

Geometry maintenance. For each selected layer, we maintain the rank-32 input-side geometry approximation defined in §4.1 and refresh the corresponding geometry statistics every $K = 50$ training steps. We observe noticeable performance degradation with larger intervals, e.g., K=200 and K=500 (Table 6 in Appendix E.1). At each refresh, we use a distributed-A (DIST-A) update: the locally accumulated input-side covariance statistics are synchronized across the data-parallel group and averaged to obtain a shared estimate of A. Each rank then locally constructs the same low-rank-plus-diagonal approximation and its Cholesky factor $L_{A,r}$. The resulting factor is held fixed until the next geometry refresh.

4.3 Geometry-Aware FP8 Gradient Communication

Given the selected layer set $\mathcal{S}$ and the current low-rank geometry factors $\{L_{A,r}^{(l)}\}_{l\in\mathcal{S}}$, GIFT applies the geometry-aware communication path only to layers in $\mathcal{S}$.

For a selected layer, let U_t denote the gradient after the low-rank input-side transformation defined in §4.1. Before FP8 quantization, we incorporate transformed-coordinate error feedback by forming $V_t = U_t + R_t$, where R_t is the residual from the previous communication step. Following standard error-feedback schemes for compressed gradient communication, the local quantization error is retained as the residual for the next step Seide et al. (2014); Karimireddy et al. (2019); Xie et al. (2024). In GIFT, this residual is maintained in the same coordinates in which FP8 quantization is performed, avoiding repeated mappings between transformed and parameter coordinates. The residual update and scale synchronization are detailed in Appendix C.3 and Appendix C.4, respectively.

5 Experimental Results

We evaluate GIFT along three dimensions: gradient communication fidelity, model-quality preservation, and end-to-end training efficiency. We use controlled FP8 baselines to isolate the effect of scaling and communication coordinates, while broader full-precision and low-precision baselines provide references for training quality and efficiency. Additional experimental settings and full results are provided in Appendix D and Appendix E.

Algorithm 1 Selective GIFT FP8 Gradient Communication

Require: Selected layers $\mathcal{S}$, low-rank geometry factors $\{L_{A,r}^{(l)}\}_{l\in\mathcal{S}}$, update interval $K = 50$

for each training step t **do**
  if $(t-1) \bmod K = 0$ **then**
    Refresh $\{L_{A,r}^{(l)}\}_{l\in\mathcal{S}}$
    Reset $\{R^{(l)}\}_{l\in\mathcal{S}} \leftarrow 0$
  end if
  for each weight gradient $W_g^{(l)}$ **do**
    if $l \in \mathcal{S}$ **then**
      $U^{(l)} \leftarrow W_g^{(l)} \big(L_{A,r}^{(l)}\big)^{-\top}$
      $V^{(l)} \leftarrow U^{(l)} + R^{(l)}$
    else
      $V^{(l)} \leftarrow W_g^{(l)}$
    end if
    $s^{(l)} \leftarrow \mathrm{SyncScale}\big(\mathrm{Scale}(V^{(l)})\big)$
    $Q^{(l)} \leftarrow \mathcal{Q}_{\mathrm{FP8}}(V^{(l)}; s^{(l)})$
    if $l \in \mathcal{S}$ **then**
      $R^{(l)} \leftarrow V^{(l)} - \mathrm{Dequantize}(Q^{(l)}; s^{(l)})$
    end if
    $\widehat{Q}^{(l)} \leftarrow \textsc{AllReduce}_{\mathrm{SUM}}(Q^{(l)})$
    $\widehat{V}^{(l)} \leftarrow \mathrm{Dequantize}(\widehat{Q}^{(l)}; s^{(l)})/N$
    if $l \in \mathcal{S}$ **then**
      $\widehat{W}_g^{(l)} \leftarrow \widehat{V}^{(l)} L_{A,r}^{(l)\top}$
    else
      $\widehat{W}_g^{(l)} \leftarrow \widehat{V}^{(l)}$
    end if
  end for
  Update parameters using $\{\widehat{W}_g^{(l)}\}$
end for

5.1 Experimental Setup

We implement GIFT in Megatron-LM Narayanan et al. (2021) and pretrain Llama-600M and Llama-2B on OpenWebText Gokaslan et al. (2019) (Appendix D.1). Following Llama pretraining settings used in prior work Zhao et al. (2024); Li et al. (2025); Pang et al. (2026), all models are trained for 60K steps with a 6K-step warmup. We use Muon Jordan et al. (2024) as the optimizer for all primary experiments (Appendix D.2).

For downstream evaluation, we use a 14-task suite consisting of BoolQ, CB, COPA, MultiRC, ReCoRD, RTE, WiC, and WSC from SuperGLUE Wang et al. (2019); LAMBADA and LAMBADA-STD Paperno et al. (2016); RACE Lai et al. (2017); M-QA Amini et al. (2019); PIQA Bisk et al. (2020); and WinoGrande Sakaguchi et al. (2020). We apply the same evaluation protocol to all methods without task-specific fine-tuning (Appendix D.4).

We perform a broad baseline comparison on Llama-600M and use Llama-2B to evaluate whether the GIFT–versus–Euclidean trends persist at a larger model scale. On Llama-600M, we compare against Muon FP32 and BF16; layer-wise and block-wise Euclidean FP8, with and without error feedback; SQ-FP8 derived from the scaling strategy of FP8-LM Peng et al. (2023); HS-FP8 derived from the scaling and smoothing strategy of SDP4Bit Jia et al. (2024); and AdamW FP32 as an optimizer reference. On Llama-2B, we compare GIFT against Muon FP32 and Layer-wise Euclidean FP8. All FP8 gradient-communication methods use the same training configuration and underlying FP8 collective communication setting. Baseline definitions are provided in Appendix D.3.

Experiments are conducted on the Vista supercomputer at the Texas Advanced Computing Center (TACC) Ruhela et al. (2024). Each node contains one NVIDIA GH200 Grace Hopper Superchip with 96 GB of HBM3 memory and uses a 400 Gb/s Mellanox NDR interconnect. Timing methodology is provided in Appendix D.5.

5.2 Geometry Ablation and Gradient Fidelity

We first examine how much reconstruction fidelity is retained as the full K-FAC transformation is simplified into the practical geometry used by GIFT. Starting from an FP32 gradient, we map it into the communication coordinates, apply an FP32→FP8→FP32 round trip, map it back to the original parameter coordinates, and compare the reconstructed gradient with its FP32 reference. To avoid relying on a single gradient snapshot, we report results averaged over five randomly sampled MLP projections from Llama-600M, with all transformation variants evaluated on the same projections. We use relative ℓ_2 error (RelL2) as the primary metric because it normalizes for differences in gradient magnitude across layers. Additional reconstruction metrics are reported in Appendix B.1.

Table 1 reveals a fidelity–complexity tradeoff. The full two-sided transformation achieves the lowest reconstruction error, reducing RelL2 by 70.7% relative to Euclidean FP8. Both one-sided transformations provide improvements, with the input-side transformation reducing RelL2 by 57.8%, versus

Table 1: FP8 reconstruction error under different communication coordinates. Lower is better.

Coordinates	RelL2 ↓	Coordinates	RelL2 ↓
Euclidean	5.272×10^{-2}	Input-side only	2.223×10^{-2}
Output-side only	2.652×10^{-2}	Low-rank input ($r = 32$)	2.494×10^{-2}
Full two-sided	1.545×10^{-2}		

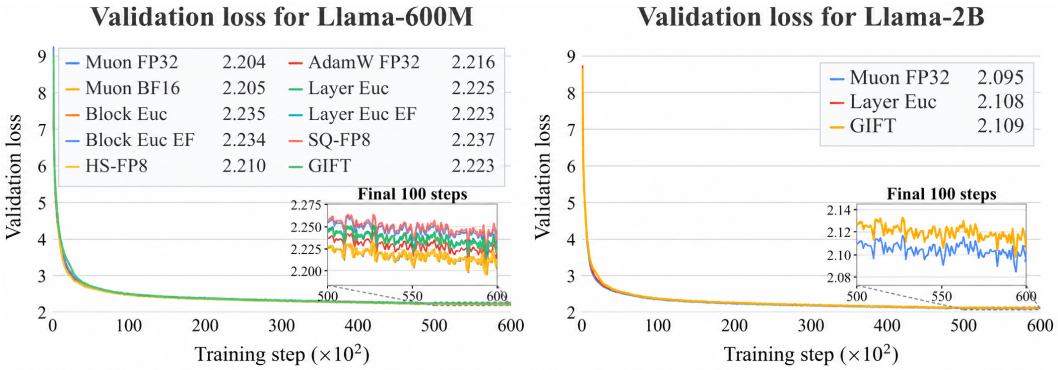


Figure 3: Validation loss for Llama-600M (left) and Llama-2B (right). Legend entries report the final validation loss at the end of training.

49.7% for the output-side transformation. This motivates the input-side design in §4.1: it provides the stronger one-sided approximation while avoiding the output-side geometry factor.

The rank-32 approximation further simplifies the input-side geometry while achieving a 52.7% reduction in RelL2 relative to Euclidean FP8. This retains approximately 91% of the error-reduction benefit of the dense input-side transformation, providing a favorable tradeoff between reconstruction fidelity and geometry complexity. Appendix B.2 compares dense, diagonal, blockwise, and lower-rank approximations.

5.3 Pretraining Convergence and Downstream Performance

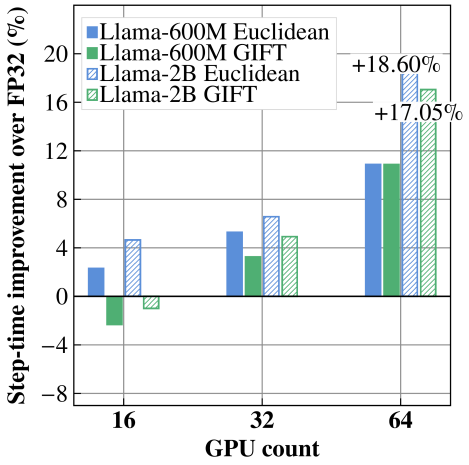
We next examine whether the improved gradient reconstruction fidelity in §5.2 translates into better preservation of model quality during pretraining. We report both validation loss and downstream-task performance because pretraining loss is primarily a convergence diagnostic and does not fully characterize downstream capability: models with similar or even identical pretraining losses can exhibit substantially different downstream performance Liu et al. (2023); Krajewski et al. (2026).

Figure 3 shows the validation-loss trajectories. Across methods, the low-precision runs remain close to the high-precision references, indicating that FP8 gradient communication does not compromise stable pretraining. The differences between communication methods are relatively small in validation loss, motivating a closer comparison of downstream capability preservation.

For downstream evaluation, we emphasize the number of tasks on which each low-precision method matches or exceeds the corresponding Muon FP32 baseline. This task-level preservation criterion complements the average score, which can obscure degradation concentrated on a subset of tasks. Figure 4(b) summarizes both metrics; complete per-task results are reported in Appendix E.1.

The downstream evaluation shows clearer differences than validation loss. On Llama-600M, GIFT matches or exceeds Muon FP32 on 8 of 14 tasks, versus 5 of 14 for Layer-wise Euclidean FP8. HS-FP8 is the only low-precision baseline with the same 8/14 count as GIFT, but has a lower average downstream score (0.5043 vs. 0.5213). Moreover, in our PyTorch-level implementation, HS-FP8 is approximately $1.8\times$ slower than GIFT and substantially slower than the FP32 baseline.

The trend holds at the larger scale. On Llama-2B, GIFT matches or exceeds Muon FP32 on 7 of 14 tasks, versus 4 of 14 for direct Layer-wise Euclidean FP8. With the reconstruction results in §5.2, this indicates that the fidelity gain from geometry-aware communication coordinates translates into stronger preservation of downstream performance, rather than merely reducing local quantization error.



(a) End-to-end step-time improvement over FP32.

Method	Avg.	≥ Muon FP32
Llama-600M		
Muon FP32	0.5060	–
AdamW FP32	0.5094	6/14
Muon BF16	0.5017	7/14
Layer-wise Euclidean FP8	0.5186	5/14
Block-wise Euclidean FP8	0.5023	4/14
Layer-wise Euclidean FP8 + EF	0.5050	3/14
Block-wise Euclidean FP8 + EF	0.5111	4/14
SQ-FP8	0.5086	6/14
HS-FP8	0.5043	8/14
GIFT	**0.5213**	**8/14**
Llama-2B		
Muon FP32	0.5102	–
Layer-wise Euclidean FP8	0.5099	4/14
GIFT	**0.5231**	**7/14**

(b) Downstream-task preservation.

Figure 4: Quality–efficiency tradeoff of low-precision gradient communication. (a) End-to-end step-time improvement over FP32 as GPU count increases. (b) Average downstream score and tasks matching or exceeding Muon FP32 for Llama-600M and Llama-2B.

5.4 End-to-End Efficiency

Finally, we evaluate whether the fidelity improvement of GIFT can be obtained while retaining the systems benefit of low-precision gradient communication. Reducing gradients from FP32 to FP8 decreases the gradient payload by 75.0%. Beyond FP8 gradients, GIFT synchronizes quantization scales at each step and periodically synchronizes the input-side covariance statistics for selected layers, from which each rank constructs a rank-32 factor. Amortized over the $K = 50$ geometry-update interval, this adds only 1.7–12 MB per step for Llama-600M and 5.2–38 MB per step for Llama-2B. Even at 2 bytes per gradient value, these amounts are around 1% or less of the corresponding full gradient volumes (approximately 1.2 GB and 4 GB, respectively). Scale synchronization adds negligible payload, so GIFT retains essentially the full $4\times$ gradient-payload reduction of FP8.

Figure 4(a) shows the end-to-end step-time improvement as the GPU count increases. At 16 GPUs, the geometry computation offsets the communication savings, making GIFT 2.33% and 1.00% slower than FP32 for Llama-600M and Llama-2B, respectively. As the GPU count increases, however, the net end-to-end benefit grows. On 64 NVIDIA GH200 Superchips, GIFT reduces end-to-end training step time by 10.87% for Llama-600M and 17.05% for Llama-2B relative to FP32 gradient communication. Layer-wise Euclidean FP8 achieves 10.88% and 18.60%, respectively; thus, at the 2B scale, GIFT is only 1.55 percentage points below direct Euclidean FP8.

At 64 GPUs, GIFT retains most of the raw FP8 speedup while providing improved gradient fidelity and downstream-task preservation. Its cost comes from geometry estimation, low-rank transformations, and transformed-coordinate error feedback on selected layers. Relative to Layer-wise Euclidean FP8, GIFT adds 4.94% and 11.07% peak memory overhead for Llama-600M and Llama-2B, respectively. Scaling and memory results appear in Appendix E.2 and Appendix E.3, respectively.

6 Conclusion

We propose GIFT to address the anisotropic gradient geometry that makes low-precision gradient communication highly direction dependent. GIFT uses K-FAC-based local geometry to construct temporary communication coordinates and reduces the full formulation to a practical input-side, rank-32 transformation applied selectively across layers. This design reduces relative FP8 gradient reconstruction error by 52.7% on average compared with Euclidean FP8 communication and improves downstream-task preservation, while retaining essentially the full 75.0% gradient-payload reduction of FP8. On 64 NVIDIA GH200 Superchips, GIFT reduces end-to-end training step time by 10.87% for Llama-600M and 17.05% for Llama-2B relative to FP32 gradient communication. These results demonstrate that adapting the communication coordinates to local gradient geometry can improve the fidelity–efficiency tradeoff of low-precision distributed training.

REPRODUCIBILITY STATEMENT

We provide detailed derivations of the geometry-informed communication coordinates in Appendix A, geometry and approximation ablations in Appendix B, implementation details in Appendix C, and complete experimental configurations, baseline definitions, evaluation protocols, and timing methodology in Appendix D. Additional downstream, scaling, and memory results are reported in Appendix E. We will publicly release the source code and experimental recipes needed to reproduce our main results. All datasets used in this work are publicly available: we pretrain on OpenWebText and evaluate on the publicly available downstream benchmarks described in Section 5.1.

AI USE STATEMENT

Generative AI tools were used to assist with implementing author-designed algorithms by generating and refining code, designing and refining scientific workflow figures, and editing the manuscript for clarity and readability. The research methodology and algorithms were developed by the authors. AI-assisted code was reviewed, tested, and revised by the authors before use, and all AI-assisted figures and manuscript edits were reviewed by the authors. No other required-disclosure uses of generative AI were involved. The authors take full responsibility for the final content of this work, including code, figures, and text produced with the aid of generative AI.

REFERENCES

Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. Qsgd: Communication-efficient sgd via gradient quantization and encoding. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://doi.org/10.48550/arXiv.1610.02132.

Shun-ichi Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10(2):251–276, 1998. URL https://doi.org/10.1162/089976698300017746.

Shun-ichi Amari. *Information Geometry and Its Applications*. Springer Tokyo, 2016. URL https://doi.org/10.1007/978-4-431-55978-8.

Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. MathQA: Towards interpretable math word problem solving with operation-based formalisms. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 2357–2367. Association for Computational Linguistics, 2019. doi: 10.18653/v1/N19-1245. URL https://aclanthology.org/N19-1245/.

Jimmy Ba, Roger B. Grosse, and James Martens. Distributed second-order optimization using kronecker-factored approximations. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=SkkTMpjex.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 7432–7439, 2020. doi: 10.1609/aaai.v34i05.6239. URL https://doi.org/10.1609/aaai.v34i05.6239.

Tim Dettmers. 8-bit approximations for parallelism in deep learning. In *International Conference on Learning Representations*, 2016. URL https://doi.org/10.48550/arXiv.1511.04561.

Runa Eschenhagen, Alexander Immer, Richard E. Turner, Frank Schneider, and Philipp Hennig. Kronecker-factored approximate curvature for modern neural network architectures. In *Advances in Neural Information Processing Systems*, volume 36, pp. 33624–33655, 2023. URL https://doi.org/10.48550/arXiv.2311.00636.

Fartash Faghri, Iman Tabrizian, Ilia Markov, Dan Alistarh, Daniel M. Roy, and Ali Ramezani-Kebrya. Adaptive gradient quantization for data-parallel sgd. In *Advances in Neural Information Processing Systems*, volume 33, pp. 3174–3185, 2020. URL https://doi.org/10.48550/arXiv.2010.12460.

Aaron Gokaslan, Vanya Cohen, Ellie Pavlick, and Stefanie Tellex. Openwebtext corpus. http://Skylion007.github.io/OpenWebTextCorpus, 2019. URL http://Skylion007.github.io/OpenWebTextCorpus.

Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1842–1850. PMLR, 2018. URL https://proceedings.mlr.press/v80/gupta18a.html.

Zhendong Huang, Hengjie Cao, Fang Dong, Ruijun Huang, Mengyi Chen, Yifeng Yang, Xin Zhang, Anrui Chen, Mingzhi Dong, Yujiang Wang, Jinlong Hou, Qin Lv, Robert P. Dick, Yuan Cheng, Fan Yang, Tun Lu, and Li Shang. Spectra: Rethinking optimizers for LLMs under spectral anisotropy. *arXiv preprint arXiv:2602.11185*, 2026. URL https://doi.org/10.48550/arXiv.2602.11185.

Jinda Jia, Cong Xie, Hanlin Lu, Daoce Wang, Hao Feng, Chengming Zhang, Baixi Sun, Haibin Lin, Zhi Zhang, Xin Liu, and Dingwen Tao. SDP4Bit: Toward 4-bit Communication Quantization in Sharded Data Parallelism for LLM Training, 2024. URL https://doi.org/10.48550/arXiv.2410.15526.

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/.

Sai Praneeth Karimireddy, Quentin Rebjock, Sebastian Stich, and Martin Jaggi. Error feedback fixes signsgd and other gradient compression schemes. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3252–3261. PMLR, 2019. URL https://proceedings.mlr.press/v97/karimireddy19a.html.

Jakub Krajewski, Amitis Shidani, Dan Busbridge, Sam Wiseman, and Jason Ramapuram. Revisiting the scaling properties of downstream metrics in large language model training. In *International Conference on Learning Representations*, 2026. URL https://doi.org/10.48550/arXiv.2512.08894.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. RACE: Large-scale reading comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 785–794. Association for Computational Linguistics, 2017. doi: 10.18653/v1/D17-1082. URL https://aclanthology.org/D17-1082/.

Minghao Li, Ran Ben Basat, Shay Vargaftik, ChonLam Lao, Kevin Xu, Michael Mitzenmacher, and Minlan Yu. THC: Accelerating distributed deep learning using tensor homomorphic compression. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*, pp. 1191–1211. USENIX Association, 2024. URL https://www.usenix.org/conference/nsdi24/presentation/li-minghao.

Siyuan Li, Juanxi Tian, Zedong Wang, Xin Jin, Zicheng Liu, Wentao Zhang, and Dan Xu. Taming llms with gradient grouping. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, pp. 3260–3279. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.164. URL https://aclanthology.org/2025.acl-long.164/.

Wenxiang Lin, Juntao Huang, Luhan Zhang, Laili Li, Xiang Bao, Mengyang Zhang, Bing Wang, and Shaohuai Shi. AGoQ: Activation and gradient quantization for memory-efficient distributed training of LLMs, 2026. URL https://doi.org/10.48550/arXiv.2605.00539.

Yujun Lin, Song Han, Huizi Mao, Yu Wang, and William J. Dally. Deep gradient compression: Reducing the communication bandwidth for distributed training. In *International Conference on Learning Representations*, 2018. URL https://doi.org/10.48550/arXiv.1712.01887.

Hong Liu, Sang Michael Xie, Zhiyuan Li, and Tengyu Ma. Same pre-training loss, better downstream: Implicit bias matters for language models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pp. 22188–22214. PMLR, 2023. URL https://doi.org/10.48550/arXiv.2210.14199.

Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization, 2019. URL https://doi.org/10.48550/arXiv.1711.05101.

Ilia Markov, Adrian Vladu, Qi Guo, and Dan Alistarh. Quantized distributed training of large models with convergence guarantees. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 24020–24044. PMLR, 2023. URL https://proceedings.mlr.press/v202/markov23a.html.

James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 2408–2417. PMLR, 2015. URL https://doi.org/10.48550/arXiv.1503.05671.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. Mixed precision training. In *International Conference on Learning Representations*, 2018. URL https://doi.org/10.48550/arXiv.1710.03740.

Paulius Micikevicius, Dusan Stosic, Neil Burgess, Marius Cornea, Pradeep Dubey, Richard Grisenthwaite, Sangwon Ha, Alexander Heinecke, Patrick Judd, John Kamalu, Naveen Mellempudi, Stuart Oberman, Mohammad Shoeybi, Michael Siu, and Hao Wu. Fp8 formats for deep learning. *arXiv preprint arXiv:2209.05433*, 2022. URL https://doi.org/10.48550/arXiv.2209.05433.

Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Anand Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, Amar Phanishayee, and Matei Zaharia. Efficient large-scale language model training on gpu clusters using megatron-lm. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2021. URL https://doi.org/10.48550/arXiv.2104.04473.

Tianyu Pang, Yujie Fang, Zihang Liu, Shenyang Deng, Lei Hsiung, Shuhua Yu, and Yaoqing Yang. Htmuon: Improving muon via heavy-tailed spectral correction. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 36504–36535. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.findings-acl.1819. URL https://aclanthology.org/2026.findings-acl.1819/.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pp. 1525–1534. Association for Computational Linguistics, 2016. doi: 10.18653/v1/P16-1144. URL https://aclanthology.org/P16-1144/.

Bowen Peng, Lizhang Chen, Baiyu Su, Jeffrey Quesnelle, Diederik P. Kingma, and Qiang Liu. Demo: Decoupled momentum optimization. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=U9oewpa7cn.

Houwen Peng, Kan Wu, Yixuan Wei, Guoshuai Zhao, Yuxiang Yang, Ze Liu, Yifan Xiong, Ziyue Yang, Bolin Ni, Jingcheng Hu, Ruihang Li, Miaosen Zhang, Chen Li, Jia Ning, Ruizhe Wang, Zheng Zhang, Shuguang Liu, Joe Chau, Han Hu, and Peng Cheng. FP8-LM: Training FP8 large language models. *arXiv preprint arXiv:2310.18313*, 2023. URL https://doi.org/10.48550/arXiv.2310.18313.

Sergio P. Perez, Yan Zhang, James Briggs, Charlie Blake, Josh Levy-Kramer, Paul Balanca, Carlo Luschi, Stephen Barlow, and Andrew Fitzgibbon. Training and inference of large language models using 8-bit floating point. *arXiv preprint arXiv:2309.17224*, 2023. URL https://doi.org/10.48550/arXiv.2309.17224.

Olivier Roy and Martin Vetterli. The effective rank: A measure of effective dimensionality. In *15th European Signal Processing Conference*, pp. 606–610, 2007. URL https://ieeexplore.ieee.org/document/7098875.

Amit Ruhela, John Cazes, John McCalpin, Carlos del Castillo-Negrete, Junjie Li, Hang Liu, Hanning Chen, Chun-Yaung Lu, Kent Milfeld, Wenyang Zhang, Ian Wang, Lars Koesterke, John DeSantis, Nic Lewis, Sean Hempel, and Dan Stanzione. Performance analysis of scientific applications on an NVIDIA grace system. In *SC24-W: Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 558–566. IEEE, 2024. doi: 10.1109/SCW63240.2024.00078. URL https://doi.org/10.1109/SCW63240.2024.00078.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 8732–8740, 2020. doi: 10.1609/aaai.v34i05.6399. URL https://doi.org/10.1609/aaai.v34i05.6399.

Frank Seide, Hao Fu, Jasha Droppo, Gang Li, and Dong Yu. 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech dnns. In *Interspeech*, pp. 1058–1062, 2014. doi: 10.21437/Interspeech.2014-274. URL https://doi.org/10.21437/Interspeech.2014-274.

Jaeyong Song, Jinkyu Yim, Jaewon Jung, Hongsun Jang, Hyung-Jin Kim, Youngsok Kim, and Jinho Lee. Optimus-CC: Efficient Large NLP Model Training with 3D Parallelism Aware Communication Compression. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, pp. 560–573. Association for Computing Machinery, 2023. ISBN 9781450399166. URL https://doi.org/10.1145/3575693.3575712.

Hanlin Tang, Shaoduo Gan, Ammar Ahmad Awan, Samyam Rajbhandari, Conglong Li, Xiangru Lian, Ji Liu, Ce Zhang, and Yuxiong He. 1-bit adam: Communication efficient large-scale training with adam's convergence speed. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 10118–10129. PMLR, 2021. URL https://doi.org/10.48550/arXiv.2102.02888.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. LLaMA: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023. URL https://doi.org/10.48550/arXiv.2302.13971.

Thijs Vogels, Sai Praneeth Karimireddy, and Martin Jaggi. Powersgd: Practical low-rank gradient compression for distributed optimization. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://doi.org/10.48550/arXiv.1905.13727.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. SuperGLUE: A stickier benchmark for general-purpose language understanding systems. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/4496bf24afe7fab6f046bf4923da8de6-Abstract.html.

Guanhua Wang, Heyang Qin, Sam Ade Jacobs, Connor Holmes, Samyam Rajbhandari, Olatunji Ruwase, Feng Yan, Lei Yang, and Yuxiong He. ZeRO++: Extremely efficient collective communication for giant model training. *arXiv preprint arXiv:2306.10209*, 2023. URL https://doi.org/10.48550/arXiv.2306.10209.

Naigang Wang, Jungwook Choi, Daniel Brand, Chia-Yu Chen, and Kailash Gopalakrishnan. Training deep neural networks with 8-bit floating point numbers. In *Advances in Neural Information Processing Systems*, volume 31, pp. 7675–7684, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/335d3d1cd7ef05ec77714a215134914c-Abstract.html.

Wei Wen, Cong Xu, Feng Yan, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Terngrad: Ternary gradients to reduce communication in distributed deep learning. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://doi.org/10.48550/arXiv.1705.07878.

Ben Williams, Kaiming Ouyang, Kamil Iskra, and Sylvain Jeaugey. Networking reliability and observability at scale with NCCL 2.24. NVIDIA Technical Blog, 2025. URL https://developer.nvidia.com/blog/networking-reliability-and-observability-at-scale-with-nccl-2-24/.

Haocheng Xi, Han Cai, Ligeng Zhu, Yao Lu, Kurt Keutzer, Jianfei Chen, and Song Han. COAT: Compressing optimizer states and activation for memory-efficient FP8 training. *arXiv preprint arXiv:2410.19313*, 2024. URL https://doi.org/10.48550/arXiv.2410.19313.

Xingyu Xie, Zhijie Lin, Kim-Chuan Toh, and Pan Zhou. Loco: Low-bit communication adaptor for large-scale model training. *arXiv preprint arXiv:2407.04480*, 2024. URL https://doi.org/10.48550/arXiv.2407.04480.

Zeke Xie, Qian-Yuan Tang, Mingming Sun, and Ping Li. On the overlooked structure of stochastic gradients. In *Advances in Neural Information Processing Systems*, volume 36, pp. 66257–66276, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/d0b2eda0386f477ab14d7e181e16c899-Abstract.html.

Hanzhen Zhao, Xingyu Xie, Cong Fang, and Zhouchen Lin. Separate: A simple low-rank projection for gradient compression in modern large-scale model training process. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=8HuLgtjqOD.

Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. Galore: Memory-efficient llm training by gradient low-rank projection. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 61121–61143. PMLR, 2024. URL https://proceedings.mlr.press/v235/zhao24s.html.

A K-FAC METRIC DERIVATION AND COMMUNICATION COORDINATES

This appendix provides the detailed derivations underlying the local K-FAC geometry and communication coordinates introduced in §3. We first decompose the full-model Fisher into layerwise blocks and derive the Kronecker-factored approximation for a linear layer. We then show how the resulting local metric is normalized into Euclidean coordinates, derive the corresponding unique dual coordinates for gradients, and finally specialize the full construction to the input-side transformation used by the practical GIFT design.

A.1 LAYERWISE FISHER STRUCTURE

Let θ denote the vector containing all model parameters. The Fisher information matrix is

$$F(\theta) = \mathbb{E}\left[g_\theta g_\theta^\top\right], \qquad g_\theta = \nabla_\theta \log p(y \mid x; \theta). \tag{9}$$

For fixed model parameters θ, $F(\theta)$ describes the local second-order geometry of the model distribution. For a small parameter perturbation $\Delta\theta$, this geometry induces the quadratic form

$$\|\Delta\theta\|^2_{F(\theta)} = \Delta\theta^\top F(\theta)\Delta\theta. \tag{10}$$

Compared with the Euclidean quantity $\|\Delta\theta\|_2^2 = \Delta\theta^\top\Delta\theta$, the Fisher weights different parameter directions differently according to their local sensitivity.

To expose the layerwise structure, partition the parameter vector and its score gradient as

$$\theta = \begin{bmatrix} \mathrm{vec}(W_1) \\ \mathrm{vec}(W_2) \\ \vdots \\ \mathrm{vec}(W_L) \end{bmatrix}, \qquad g_\theta = \begin{bmatrix} g_{W_1} \\ g_{W_2} \\ \vdots \\ g_{W_L} \end{bmatrix}, \tag{11}$$

where

$$g_{W_l} = \nabla_{\mathrm{vec}(W_l)} \log p(y \mid x; \theta). \tag{12}$$

Substituting this partition into Equation 9 gives the block matrix

$$F(\theta) = \begin{bmatrix} \mathbb{E}[g_{W_1} g_{W_1}^\top] & \mathbb{E}[g_{W_1} g_{W_2}^\top] & \cdots \\ \mathbb{E}[g_{W_2} g_{W_1}^\top] & \mathbb{E}[g_{W_2} g_{W_2}^\top] & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix}. \tag{13}$$

The diagonal blocks describe within-layer geometry, whereas the off-diagonal blocks describe interactions between parameters belonging to different layers. Define the Fisher block associated with layer l as

$$F_{W_l} = \mathbb{E}\left[g_{W_l} g_{W_l}^\top\right]. \tag{14}$$

A layerwise block-diagonal approximation discards the cross-layer blocks, yielding

$$F(\theta) \approx \mathrm{blockdiag}\left(F_{W_1}, F_{W_2}, \ldots, F_{W_L}\right). \tag{15}$$

Under this approximation, the full-model quadratic form decomposes into a sum of independent layerwise contributions,

$$\|\Delta\theta\|^2_{F(\theta)} \approx \sum_{l=1}^{L} \mathrm{vec}(\Delta W_l)^\top F_{W_l} \,\mathrm{vec}(\Delta W_l). \tag{16}$$

This decomposition allows the local communication geometry of each layer to be constructed independently rather than requiring a single full-model Fisher matrix.

A.2 KRONECKER FACTORIZATION FOR A LINEAR LAYER

We now derive a tractable approximation for one of the layerwise blocks F_W. Consider a linear layer

$$s = Wa, \tag{17}$$

where $W \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, $a \in \mathbb{R}^{d_{\text{in}}}$ is the layer input, and $s \in \mathbb{R}^{d_{\text{out}}}$ is the pre-activation output.

Let

$$\delta = \frac{\partial \mathcal{L}}{\partial s} \in \mathbb{R}^{d_{\text{out}}} \tag{18}$$

denote the backpropagated gradient with respect to s. For a negative-log-likelihood objective, $\mathcal{L} = -\log p(y \mid x; \theta)$, the loss gradient differs from the score gradient only by a sign. Because the Fisher is formed from an outer product, this sign cancels. We can therefore express the corresponding layerwise Fisher block using the loss gradient.

For an individual weight element W_{ij},

$$\frac{\partial \mathcal{L}}{\partial W_{ij}} = \frac{\partial \mathcal{L}}{\partial s_i} \frac{\partial s_i}{\partial W_{ij}} = \delta_i a_j. \tag{19}$$

Collecting these derivatives gives

$$\nabla_W \mathcal{L} = \delta a^\top. \tag{20}$$

The Fisher block is defined over the vectorized parameters of W. Under column-major vectorization,

$$\text{vec}(uv^\top) = v \otimes u \tag{21}$$

gives

$$\text{vec}\left(\nabla_W \mathcal{L}\right) = \text{vec}(\delta a^\top) = a \otimes \delta. \tag{22}$$

Let

$$g_W = \text{vec}(\nabla_W \mathcal{L}). \tag{23}$$

The exact layerwise Fisher block is then

$$\begin{aligned} F_W &= \mathbb{E}\left[g_W g_W^\top\right] \\ &= \mathbb{E}\left[(a \otimes \delta)(a \otimes \delta)^\top\right]. \end{aligned} \tag{24}$$

Using

$$(a \otimes \delta)^\top = a^\top \otimes \delta^\top \tag{25}$$

and

$$(A \otimes B)(C \otimes D) = (AC) \otimes (BD), \tag{26}$$

we obtain

$$\begin{aligned} (a \otimes \delta)(a \otimes \delta)^\top &= (a \otimes \delta)(a^\top \otimes \delta^\top) \\ &= (aa^\top) \otimes (\delta\delta^\top). \end{aligned} \tag{27}$$

Therefore,

$$F_W = \mathbb{E}\left[(aa^\top) \otimes (\delta\delta^\top)\right]. \tag{28}$$

K-FAC approximates the joint second-order statistics by separating the two factors,

$$\mathbb{E}\left[(aa^\top) \otimes (\delta\delta^\top)\right] \approx \mathbb{E}[aa^\top] \otimes \mathbb{E}[\delta\delta^\top]. \tag{29}$$

Defining

$$A = \mathbb{E}[aa^\top] \in \mathbb{R}^{d_{\text{in}} \times d_{\text{in}}}, \qquad G = \mathbb{E}[\delta\delta^\top] \in \mathbb{R}^{d_{\text{out}} \times d_{\text{out}}}, \tag{30}$$

gives

$$F_W \approx A \otimes G. \tag{31}$$

For the remainder of the derivation, denote the K-FAC approximate metric by

$$\overline{F}_W := A \otimes G. \tag{32}$$

The approximation is therefore $F_W \approx \overline{F}_W$, while the subsequent coordinate identities hold exactly with respect to $\overline{F}_W$.

A.3 Metric Normalization in Matrix Coordinates

For a perturbation $\Delta W \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, the quadratic form induced by the K-FAC metric is

$$\|\Delta W\|^2_{\overline{F}_W} = \operatorname{vec}(\Delta W)^\top (A \otimes G) \operatorname{vec}(\Delta W). \tag{33}$$

For matrices of compatible dimensions,

$$\operatorname{vec}(X)^\top (B \otimes C) \operatorname{vec}(X) = \operatorname{tr}\left(X^\top C X B^\top\right). \tag{34}$$

Because A and G are symmetric,

$$\begin{aligned} \|\Delta W\|^2_{\overline{F}_W} &= \operatorname{tr}\left(\Delta W^\top G \Delta W A\right) \\ &= \operatorname{tr}\left(A \Delta W^\top G \Delta W\right). \end{aligned} \tag{35}$$

After damping or regularization, let

$$A = L_A L_A^\top, \qquad G = L_G L_G^\top, \tag{36}$$

with $A, G \succ 0$. Substituting these factorizations gives

$$\begin{aligned} \|\Delta W\|^2_{\overline{F}_W} &= \operatorname{tr}\left(L_A L_A^\top \Delta W^\top L_G L_G^\top \Delta W\right) \\ &= \operatorname{tr}\left[\left(L_G^\top \Delta W L_A\right)^\top \left(L_G^\top \Delta W L_A\right)\right] \\ &= \left\|L_G^\top \Delta W L_A\right\|_F^2. \end{aligned} \tag{37}$$

Hence the parameter-coordinate map

$$Z = \mathcal{T}_p(\Delta W) = L_G^\top \Delta W L_A \tag{38}$$

satisfies

$$\|\Delta W\|^2_{\overline{F}_W} = \|Z\|_F^2. \tag{39}$$

Therefore, every constant-metric contour

$$\|\Delta W\|^2_{\overline{F}_W} = c \tag{40}$$

is mapped to

$$\|Z\|_F^2 = c. \tag{41}$$

Thus, the ellipsoidal contours induced by the local K-FAC metric become Euclidean spheres in the transformed parameter coordinates. Since $F_W \approx \overline{F}_W$, this yields the approximate metric normalization stated in the main text,

$$\|\Delta W\|^2_{F_W} \approx \|\mathcal{T}_p(\Delta W)\|_F^2. \tag{42}$$

A.4 Dual Coordinates for Gradient Communication

The normalized coordinates above describe parameter perturbations. Gradients are dual objects, so their coordinates must preserve their first-order action on perturbations.

Let

$$W_g = \nabla_W \mathcal{L}. \tag{43}$$

Under the Frobenius inner product,

$$d\mathcal{L}(W)[\Delta W] = \langle W_g, \Delta W \rangle_F = \operatorname{tr}\left(W_g^\top \Delta W\right). \tag{44}$$

Using

$$Z = L_G^\top \Delta W L_A, \tag{45}$$

we recover the original perturbation as

$$\Delta W = L_G^{-\top} Z L_A^{-1}. \tag{46}$$

Substituting this expression into the differential gives

$$\begin{aligned} d\mathcal{L}(W)[\Delta W] &= \operatorname{tr}\left(W_g^\top L_G^{-\top} Z L_A^{-1}\right) \\ &= \operatorname{tr}\left(L_A^{-1} W_g^\top L_G^{-\top} Z\right) \\ &= \operatorname{tr}\left[\left(L_G^{-1} W_g L_A^{-\top}\right)^\top Z\right] \\ &= \left\langle L_G^{-1} W_g L_A^{-\top}, Z\right\rangle_F . \end{aligned} \tag{47}$$

The corresponding gradient coordinates are therefore

$$\widetilde{W}_g = \mathcal{T}_g(W_g) = L_G^{-1} W_g L_A^{-\top}, \tag{48}$$

for which

$$\langle W_g, \Delta W\rangle_F = \left\langle \widetilde{W}_g, Z\right\rangle_F . \tag{49}$$

It remains to show that this dual map is unique. Suppose another linear map $\mathcal{S}_g$ also preserves the pairing for every W_g and ΔW:

$$\langle W_g, \Delta W\rangle_F = \left\langle \mathcal{S}_g(W_g), L_G^\top \Delta W L_A\right\rangle_F . \tag{50}$$

Since L_A and L_G are invertible, $Z = L_G^\top \Delta W L_A$ ranges over the entire transformed matrix space. Combining Equation 49 and Equation 50 therefore gives

$$\left\langle \mathcal{S}_g(W_g) - L_G^{-1} W_g L_A^{-\top}, Z\right\rangle_F = 0 \qquad \text{for all } Z. \tag{51}$$

By nondegeneracy of the Frobenius inner product,

$$\mathcal{S}_g(W_g) = L_G^{-1} W_g L_A^{-\top}. \tag{52}$$

Hence Equation 48 is the unique linear gradient coordinate map preserving the gradient–perturbation pairing.

After low-precision communication produces $\widehat{\widetilde{W}}_g$, the inverse dual transformation is

$$\widehat{W}_g = L_G \widehat{\widetilde{W}}_g L_A^\top, \tag{53}$$

which returns the communicated gradient to the original parameter coordinates before the optimizer update.

A.5 Input-Side Specialization

The derivation above yields the full two-sided K-FAC communication coordinates. The practical GIFT design retains only the input-side geometry, motivated by the reconstruction experiments in Appendix B.1.

Removing the output-side geometry corresponds to setting

$$G = I, \qquad L_G = I. \tag{54}$$

Substituting $G = I$ into Equation 35 gives

$$\begin{aligned} \|\Delta W\|_A^2 &= \operatorname{tr}\left(A \Delta W^\top \Delta W\right) \\ &= \operatorname{tr}\left[(\Delta W L_A)^\top (\Delta W L_A)\right] \\ &= \|\Delta W L_A\|_F^2 . \end{aligned} \tag{55}$$

Thus the parameter perturbation coordinates reduce to

$$Z_A = \Delta W L_A. \tag{56}$$

Applying $L_G = I$ to the full dual transformation in Equation 48 gives

$$\widetilde{W}_g = W_g L_A^{-\top}. \tag{57}$$

This result can also be verified directly from pairing invariance:

$$\begin{aligned}\langle W_g, \Delta W\rangle_F &= \operatorname{tr}\left(W_g^\top Z_A L_A^{-1}\right)\\ &= \operatorname{tr}\left[\left(W_g L_A^{-\top}\right)^\top Z_A\right]\\ &= \left\langle \widetilde{W}_g, Z_A\right\rangle_F .\end{aligned} \tag{58}$$

After communication in the input-side coordinates, the synchronized gradient is reconstructed by

$$\widehat{W}_g = \widehat{\widetilde{W}}_g L_A^\top. \tag{59}$$

The resulting communication path is

$$W_g \;\longrightarrow\; W_g L_A^{-\top} \;\longrightarrow\; \widehat{\widetilde{W}}_g \;\longrightarrow\; \widehat{\widetilde{W}}_g L_A^\top. \tag{60}$$

This input-side transformation forms the theoretical starting point of the practical GIFT communication path. Its low-rank approximation is introduced in §4.1, while selective application across layers is described in §4.2; additional implementation details are provided in Appendix C.

B Geometry Analysis and Approximation Ablations

This appendix provides the reconstruction experiments supporting the practical geometry approximations introduced in §4.1. We first examine whether both sides of the full K-FAC-based transform are necessary, and then evaluate alternative representations of the remaining input-side geometry.

B.1 Full versus One-Sided Geometry

We evaluate each communication transform through a controlled one-step FP8 round-trip experiment. Starting from an FP32 gradient, we map it into the corresponding communication coordinates, apply an FP32→FP8→FP32 round trip, transform the result back to the original parameter coordinates, and compare the reconstructed gradient with the original FP32 gradient.

To reduce sensitivity to any single gradient snapshot, we uniformly sample five MLP projections without replacement from the 64 MLP projections of Llama-600M using a fixed random seed. At the first optimizer step of a freshly initialized model, we collect the FP32 weight gradient together with the corresponding input- and output-side covariance statistics for each sampled projection. All transformation variants are evaluated on the same five projections, and we report the arithmetic mean across them.

We report

$$\begin{aligned}\mathrm{RelL2} &= \frac{\|\hat{g}-g\|_2}{\|g\|_2},\\ \mathrm{Cos} &= \frac{\langle \hat{g}, g\rangle}{\|\hat{g}\|_2\|g\|_2},\\ \mathrm{MaxErr} &= \|\hat{g}-g\|_\infty,\\ \mathrm{MSE} &= \frac{\|\hat{g}-g\|_2^2}{n},\end{aligned} \tag{61}$$

where g is the original FP32 gradient, $\hat{g}$ is the reconstructed gradient, and n is the number of tensor elements. We use RelL2 as the primary reconstruction metric because it normalizes for differences in gradient magnitude across projections, while Cos, MaxErr, and MSE provide complementary measures of directional and absolute reconstruction error.

For the comparison below, the full two-sided transform uses both K-FAC factors, while the one-sided variants retain only the input-side or output-side factor.

Both one-sided transformations substantially improve reconstruction relative to Euclidean FP8. Averaged across the five sampled projections, the output-side transform reduces RelL2 by 49.7%, while

Table 2: FP8 round-trip error for the full and one-sided geometry transforms, averaged over five randomly sampled MLP projections. Lower is better except for Cos.

Method	RelL2	Cos	MaxErr	MSE
Euclidean baseline	5.271663e-02	9.986095e-01	3.467217e-04	2.973700e-10
Output-side only	2.652068e-02	9.996487e-01	2.563458e-04	8.128108e-11
Input-side only	2.223073e-02	9.997515e-01	1.712002e-04	5.202451e-11
Full two-sided	1.544876e-02	9.998816e-01	1.063159e-04	2.253723e-11

the input-side transform achieves a larger 57.8% reduction. The full two-sided transform provides the strongest reconstruction fidelity, reducing RelL2 by 70.7%.

These results show that both K-FAC factors contain useful geometric information, but the input-side factor provides the stronger of the two one-sided approximations. We therefore retain the input-side geometry in the practical design, avoiding the need to maintain and apply the output-side factor while retaining approximately 82% of the RelL2-reduction benefit of the full two-sided transformation.

B.2 Approximating the Input-Side Geometry

After retaining only the input-side geometry, we next examine how accurately this factor must be represented. We compare dense, diagonal, blockwise, and low-rank representations using the same FP8 round-trip protocol. The low-rank construction is defined in §4.1; additional implementation details are provided in Appendix C.1.

Table 3: FP8 round-trip error for different approximations of the input-side geometry, averaged over five randomly sampled MLP projections. Rank 32 is the operating point used in the main experiments.

Method	RelL2	Cos	MaxErr	MSE
Euclidean baseline	5.271663e-02	9.986095e-01	3.467217e-04	2.973700e-10
Input-side full-A	2.223073e-02	9.997515e-01	1.712002e-04	5.202451e-11
Input-side diag-A	5.269214e-02	9.986108e-01	3.486778e-04	2.971738e-10
Input-side block-A-32	4.454952e-02	9.990010e-01	3.403722e-04	2.408845e-10
Input-side block-A-64	4.099022e-02	9.991543e-01	3.131287e-04	2.037892e-10
Input-side block-A-128	3.687698e-02	9.993176e-01	3.099280e-04	1.608426e-10
Input-side low-rank-A-8	2.765272e-02	9.996106e-01	1.704906e-04	7.650882e-11
Input-side low-rank-A-16	2.642331e-02	9.996447e-01	1.718975e-04	6.695703e-11
Input-side low-rank-A-32	2.493969e-02	9.996852e-01	1.686031e-04	6.084948e-11

A diagonal approximation provides little improvement over Euclidean FP8, showing that independent per-dimension scaling does not capture the useful input-side correlations. Blockwise approximations improve as the block size increases, but remain noticeably worse than the dense input-side transform.

Low-rank approximations provide a substantially stronger approximation of the input-side geometry. Reconstruction fidelity improves with rank, with rank 32 achieving the lowest RelL2 and MSE and the highest cosine similarity among the evaluated reduced representations. Its mean RelL2 of 2.494×10^{-2} corresponds to a 52.7% reduction relative to Euclidean FP8, compared with a 57.8% reduction for the dense input-side transform. Thus, the rank-32 approximation retains approximately 91% of the RelL2-reduction benefit of the dense input-side geometry while using a substantially simplified representation. We therefore use rank 32 as the operating point in the main experiments.

B.3 Layer-Wise Geometry Across MLP Projections

To verify that the geometry illustrated in Figure 1 is not specific to a single projection, we compute the same geometry statistics for all 64 MLP projections of Llama-600M, comprising both FC1 and FC2 projections from each of the 32 layers. The projections are ordered layer by layer, with FC1 followed by FC2 within each layer. For each projection, we measure normalized effective rank (NERank) and the fraction of spectral energy in the leading singular direction (Top-1) in both Euclidean and geometry-informed coordinates.

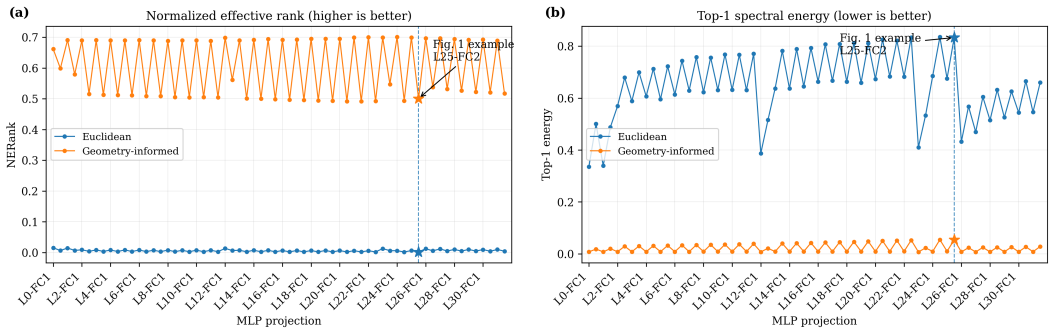


Figure 5: Geometry statistics across all 64 MLP projections of Llama-600M, including both FC1 and FC2 projections from each of the 32 layers. Projections are ordered layer by layer as FC1 followed by FC2. For readability, only every fourth projection is labeled on the x-axis; these displayed tick labels therefore correspond to FC1 projections, while the intervening unlabeled points include the corresponding FC2 projections as well as the remaining FC1/FC2 projections. Geometry-informed coordinates increase normalized effective rank (left; higher is better) and reduce Top-1 spectral energy (right; lower is better). Layer 25 FC2, used for the visualization in Fig. 1, is highlighted.

As shown in Figure 5, geometry-informed coordinates consistently increase NERank and reduce Top-1 spectral concentration across the MLP projections, indicating that the anisotropy reduction illustrated in Figure 1 is not confined to a single example. The projection visualized in Figure 1, Layer 25 FC2, is highlighted; it exhibits the largest relative increase in NERank among the 64 projections.

C Additional GIFT Implementation Details

This appendix provides additional implementation details for the low-rank geometry update, automatic layer selection, transformed-coordinate error feedback, and scale synchronization used by GIFT.

C.1 Low-Rank Geometry Update

The low-rank input-side geometry used by GIFT is defined in §4.1. In all main experiments, we use rank $r = 32$ and refresh the corresponding geometry statistics every $K = 50$ training steps.

For each selected layer, we first aggregate the input-side covariance statistics across the data-parallel group. We then retain the top-r eigenspace of the synchronized covariance and preserve the diagonal residual outside this subspace. Specifically, after removing the rank-r spectral component, the remaining diagonal entries are lower-bounded by a damping value of 10^{-3} to ensure numerical stability. The resulting low-rank-plus-diagonal approximation is factorized using a lower-triangular Cholesky factor $L_{A,r}$.

The forward communication transform $W_g L_{A,r}^{-\top}$ is implemented using a triangular solve rather than explicitly forming $L_{A,r}^{-1}$. The resulting factor is held fixed between geometry refreshes and used for the forward transformation and inverse mapping,

$$\widetilde{W}_g = W_g L_{A,r}^{-\top}, \qquad \widehat{W}_g = \widehat{\widetilde{W}}_g L_{A,r}^{\top}. \tag{62}$$

Appendix B.2 compares the rank-32 configuration against dense, diagonal, blockwise, and low-rank alternatives.

C.2 Automatic Layer Selection

Before formal pretraining, GIFT performs a separate $T_{\text{prof}} = 200$-step Layer-wise Euclidean FP8 profiling run over candidate MLP layers. For each candidate layer l, we count the number of quantized gradient values that reach the positive or negative FP8 representable boundary at profiling step

t, denoted by $v_l^{(t)}$. The cumulative saturation score of layer l is

$$v_l = \sum_{t=1}^{T_{\text{prof}}} v_l^{(t)}. \tag{63}$$

We also accumulate the quantization error of each layer over the profiling run, denoted by e_l.

After profiling, candidate layers are sorted in descending order of v_l. Let $\pi(1), \pi(2), \ldots$ denote this ordering. GIFT selects the smallest prefix of layers whose cumulative quantization error accounts for at least a fraction $\rho = 0.3$ of the total quantization error:

$$k^\star = \min \left\{ k : \frac{\sum_{i=1}^{k} e_{\pi(i)}}{\sum_{l \in \mathcal{L}_{\text{cand}}} e_l} \geq \rho \right\}, \qquad \rho = 0.3. \tag{64}$$

The selected layer set is

$$\mathcal{S} = \{\pi(1), \ldots, \pi(k^\star)\}. \tag{65}$$

The profiling run is used only to determine $\mathcal{S}$. Formal pretraining then starts from a fresh model initialization, and the selected set remains fixed throughout that run.

C.3 Transformed-Coordinate Error Feedback

For a selected layer, let

$$U_t = W_{g,t} L_{A,r}^{-\top} \tag{66}$$

denote the gradient in the geometry-aware coordinates, and let R_t denote the residual from the previous communication step. The geometry factor $L_{A,r}$ is held fixed within each $K = 50$-step refresh interval. At the beginning of a new interval, when $L_{A,r}$ is refreshed, we reset R_t to zero before applying the updated transform.

Before FP8 quantization, GIFT forms

$$V_t = U_t + R_t. \tag{67}$$

Using the synchronized communication scale s_t, the residual is updated from the local quantization error:

$$R_{t+1} = V_t - \text{Dequantize}\left(\mathcal{Q}_{\text{FP8}}(V_t; s_t); s_t\right). \tag{68}$$

Thus, within each refresh interval, the gradient and residual remain in the same transformed coordinate system. Resetting the residual when the geometry is refreshed prevents quantization errors from being carried across different coordinate systems.

C.4 Scale Synchronization

We use the FP8 E5M2 format for gradient communication. Before quantization, each participating data-parallel rank computes its local scaling statistic from the maximum absolute value of the transformed tensor. We synchronize this statistic across ranks using a MAX reduction,

$$s = \text{SyncScale}\left(s_{\text{local}}\right), \tag{69}$$

so that all ranks use a consistent communication scale.

The synchronized scale is used for FP8 quantization, collective communication, dequantization, and the local transformed-coordinate error-feedback update. This ensures that all ranks participating in the collective interpret the FP8 representation using the same scale.

D Additional Experimental Details

D.1 Model Configurations

We evaluate GIFT using two decoder-only Llama-style Transformer configurations with approximately 600M and 2B parameters. Table 4 summarizes the model architectures. The same model configuration is used for all communication methods at each model scale.

Table 4: Model configurations used in our experiments.

Configuration	∼600M	∼2B
Number of layers	32	24
Hidden size	1280	2560
Attention heads	20	20
FFN hidden size	3456	6912
Sequence length	2048	2048

D.2 Training Configuration

All methods at a given model scale use the same model architecture, dataset, batching configuration, learning-rate schedule, and training horizon. OpenWebText Gokaslan et al. (2019) is used for pretraining.

Muon Jordan et al. (2024) is used for the primary controlled experiments, while AdamW FP32 is included as an optimizer reference. Unless otherwise specified, changing the gradient-communication method does not modify the model architecture, data ordering, optimizer hyperparameters, or parallelization configuration.

Table 5 summarizes the principal training hyperparameters.

Table 5: Training configuration used for both Llama-600M and Llama-2B.

Configuration	Value
Sequence length	2048
Global batch size	512
Micro batch size	8
Optimizer	Muon
Learning rate	5×10^{-4}
Minimum learning rate	5×10^{-5}
LR schedule	Cosine
Warmup steps	6,000
Weight decay	0.1
Gradient clipping	1.0
Training steps	60,000

D.3 Baseline Definitions

We evaluate the following full- and low-precision baselines.

Muon FP32. Muon training with FP32 gradient communication. This serves as the primary high-precision reference for the controlled communication experiments.

Muon BF16. Muon training with BF16 gradient communication.

Layer-wise Euclidean FP8. Gradients are directly scaled and quantized in their original Euclidean coordinates using the configured layer-wise scaling granularity.

Block-wise Euclidean FP8. Each gradient is partitioned into blocks of 512 elements, with an independent FP8 scale computed for each block.

Layer-wise Euclidean FP8 + Error Feedback. A GIFT ablation that augments layer-wise Euclidean FP8 communication with residual error compensation in the original gradient coordinates. This baseline isolates the effect of error feedback without geometry-aware coordinate transformation.

Block-wise Euclidean FP8 + Error Feedback. A GIFT ablation that augments block-wise Euclidean FP8 communication, using blocks of 512 elements, with residual error compensation in the

original gradient coordinates. This baseline isolates the effect of error feedback under finer-grained Euclidean scaling without geometry-aware coordinate transformation.

SQ-FP8. SQ-FP8 adapts the scaling strategy used by FP8-LM Peng et al. (2023) to our FP8 gradient-communication setting. It is a controlled scaling baseline and does not reproduce the complete FP8-LM training system.

HS-FP8. HS-FP8 adapts the scaling and smoothing strategy motivated by SDP4Bit Jia et al. (2024) to our FP8 gradient-communication setting. It uses the same FP8 communication setting as the other controlled baselines and does not reproduce the complete SDP4Bit system.

AdamW FP32. We additionally report FP32 training with AdamW Loshchilov & Hutter (2019) as an optimizer-level reference to prior low-precision LLM training studies.

GIFT. GIFT uses the selective geometry-aware FP8 communication path described in §3 and §4.

D.4 Downstream Evaluation

We evaluate pretrained checkpoints on 14 downstream tasks using the same evaluation protocol for every communication method. No task-specific fine-tuning is performed. For each task, we report the standard evaluation metric used by the benchmark.

In addition to individual task scores, we report the number of tasks on which a method matches or exceeds the corresponding Muon FP32 result. This task-level preservation count complements aggregate performance because a similar average score can otherwise hide degradation concentrated on a subset of tasks.

Complete per-task results for both model scales are reported in Appendix E.1.

D.5 System and Timing Methodology

All end-to-end timing experiments are conducted on NVIDIA GH200 Superchips on the Vista system at the Texas Advanced Computing Center (TACC). For a fixed model and GPU count, all compared methods use the same batching, optimizer, and parallelism configurations.

End-to-end time includes model computation, optimizer updates, quantization and dequantization, collective communication, synchronization, and any method-specific transformation overhead.

In addition to end-to-end timing, we account for the average communicated data volume per training step, including method-specific auxiliary communication. Relative to FP32 gradient communication, Layer-wise Euclidean FP8 reduces the communicated gradient volume by 75.0%.

For GIFT, selected layers additionally synchronize the input-side covariance statistics once every $K = 50$ training steps, from which each rank locally constructs the rank-32 geometry factor. Amortized over the update interval, this adds only 1.7–12 MB per step for Llama-600M and 5.2–38 MB per step for Llama-2B, depending on the dimensions of the selected MLP projections. Scale synchronization contributes negligible additional communication. Thus, the auxiliary communication remains small relative to the gradient volume, and GIFT retains nearly the full communication-volume reduction provided by FP8.

E Additional Experimental Results

E.1 Full Downstream Results

Tables 6 and 7 report the complete task-level downstream results for Llama-600M and Llama-2B, respectively. Higher values are better. The final column counts the number of tasks on which each method matches or exceeds the corresponding Muon FP32 baseline.

Table 6: Full downstream evaluation of Llama-600M. Higher is better; the final column counts tasks matching or exceeding Muon FP32.

Method	BOOLQ	CB-ACC	COPA	MUL-RC	RCD-F1	RTE	WiC	WSC	LAMB.	RACE	M-QA	PIQA	WINO.	LAMB-STD	Avg.	≥ FP32
Muon FP32	0.5908	0.2679	0.7500	0.5532	0.8352	0.5090	0.5000	0.4423	0.4941	0.3120	0.2251	0.6719	0.5612	0.3716	0.5060	–
AdamW FP32	0.5385	0.1786	0.7100	0.5503	0.8253	0.5632	0.5000	0.6442	0.4885	0.3244	0.2265	0.6681	0.5375	0.3771	0.5094	6/14
Muon BF16	0.6058	0.2321	0.7100	0.5608	0.8346	0.5307	0.4875	0.4231	0.4865	0.3158	0.2295	0.6806	0.5391	0.3873	0.5017	7/14
Layer-wise Euclidean FP8	0.6119	0.4107	0.7100	0.5025	0.8271	0.5199	0.4969	0.5865	0.4824	0.3100	0.2201	0.6659	0.5320	0.3850	0.5186	5/14
Block-wise Euclidean FP8	0.5679	0.1250	0.7100	0.5549	0.8205	0.5487	0.5188	0.6058	0.4795	0.3024	0.2161	0.6665	0.5478	0.3685	0.5023	4/14
Layer-wise Euclidean FP8 + EF	0.5976	0.3929	0.6700	0.5487	0.8288	0.5415	0.4984	0.3750	0.4939	0.3110	0.2245	0.6659	0.5517	0.3703	0.5050	3/14
Block-wise Euclidean FP8 + EF	0.6040	0.3929	0.7300	0.5559	0.8252	0.5487	0.4734	0.4327	0.4795	0.3100	0.2235	0.6654	0.5438	0.3710	0.5111	4/14
SQ-FP8	0.5630	0.2857	0.6700	0.5155	0.8308	0.5162	0.4859	0.6250	0.4801	0.3129	0.2255	0.6708	0.5556	0.3839	0.5086	6/14
HS-FP8	0.5667	0.0893	0.7200	0.5448	0.8359	0.5523	0.5047	0.6058	0.4993	0.3033	0.2268	0.6752	0.5588	0.3771	0.5043	8/14
GIFT ($K = 200$)	0.5994	0.0893	0.6600	0.5437	0.8267	0.5271	0.5110	0.5865	0.4799	0.3120	0.2221	0.6757	0.5233	0.3679	0.4946	6/14
GIFT ($K = 500$)	0.6278	0.1607	0.6700	0.4251	0.8245	0.5162	0.4937	0.4231	0.4879	0.3043	0.2214	0.6638	0.5320	0.3747	0.4804	3/14
GIFT	0.6043	0.4107	0.7100	0.5103	0.8268	0.5487	0.5000	0.5673	0.4824	0.3225	0.2241	0.6719	0.5430	0.3757	0.5213	8/14

Table 7: Full downstream evaluation of Llama-2B. Higher is better; the final column counts tasks matching or exceeding Muon FP32.

Method	BOOLQ	CB-ACC	COPA	MUL-RC	RCD-F1	RTE	WiC	WSC	LAMB.	RACE	M-QA	PIQA	WINO.	LAMB-STD	Avg.	≥ FP32
Muon FP32	0.5780	0.1786	0.7700	0.5336	0.8556	0.5090	0.5000	0.3942	0.5476	0.3340	0.2335	0.7002	0.5683	0.4395	0.5102	–
Layer-wise Euclidean FP8	0.4869	0.0893	0.7600	0.5091	0.8524	0.5343	0.4781	0.6538	0.5389	0.3359	0.2241	0.6926	0.5375	0.4460	0.5099	4/14
GIFT	0.5783	0.2500	0.7400	0.4462	0.8593	0.5379	0.5157	0.6154	0.5348	0.3292	0.2268	0.6975	0.5730	0.4186	0.5231	7/14

E.2 Multi-GPU Scaling Results

Table 8 reports the raw end-to-end timing measurements used to construct the scaling results in §5.4. Step-time improvement is measured relative to the corresponding FP32 gradient-communication baseline at the same model and GPU scale.

Table 8: End-to-end step-time improvement (%) over FP32 gradient communication across model and GPU scales. Positive values indicate faster training than FP32.

Model	Method	16 GPUs	32 GPUs	64 GPUs
Llama-600M	Layer-wise Euclidean FP8	2.31	5.29	10.88
	GIFT	−2.33	3.26	10.87
Llama-2B	Layer-wise Euclidean FP8	4.65	6.57	18.60
	GIFT	−1.00	4.93	17.05

The difference between the 75.0% reduction in gradient payload and the smaller end-to-end speedup arises because total training step time also includes forward and backward computation, optimizer updates, geometry transformations, quantization and dequantization, synchronization, and communication unrelated to the compressed gradient payload.

E.3 Memory Overhead

Table 9 reports the measured peak GPU memory usage of FP32 training, direct Layer-wise Euclidean FP8 communication, and GIFT, together with the memory increase relative to the FP32 baseline. All measurements are obtained on 64 NVIDIA GH200 GPUs with a global batch size of 512 and a micro-batch size of 8.

Table 9: Peak GPU memory usage and overhead relative to FP32 training.

Model	FP32	Layer-wise Euclidean FP8	GIFT
Llama-600M	55.28 GiB	57.23 GiB (+3.52%)	60.06 GiB (+8.63%)
Llama-2B	81.61 GiB	83.60 GiB (+2.44%)	92.86 GiB (+13.78%)

Across both model sizes, the peak-memory overhead of GIFT remains below 14% relative to FP32 training.