Mixtures of Subspaces for Bandwidth Efficient Context Parallel Training

Sameera Ramasinghe **Ajanthan Thalaiyasingam** **Hadi Mohaghegh Dolatabadi**

Gil Avraham **Violetta Shevchenko** **Yan Zuo** **Chamin Hewa Koneputugodage**

Alexander Long

Pluralis Research

Abstract

Pretraining language models with extended context windows enhances their ability to leverage rich information during generation. Existing methods split input sequences into chunks, broadcast them across multiple devices, and compute attention block by block which incurs significant communication overhead. While feasible in high-speed clusters, these methods are impractical for decentralized training over low-bandwidth connections. We propose a compression method for communication-efficient context parallelism in decentralized settings, achieving a remarkable compression rate of over 95% with negligible overhead and no loss in convergence. Our key insight is to exploit the intrinsic low-rank structure of activation outputs by dynamically constraining them to learned mixtures of subspaces via efficient reparameterizations. We demonstrate scaling billion-parameter decentralized models to context lengths exceeding 100K tokens on networks as slow as 300Mbps, matching the wall-clock convergence speed of centralized models on 100Gbps interconnects.

1 Introduction

Rapid scaling of large language models (LLMs) has made distributed training a necessity [23, 20, 11, 38]. As both model size and context length continue to grow, efficient training increasingly depends on parallelization across multiple devices. Traditional distributed training paradigms assume high-bandwidth, low-latency interconnects, typically available in centralized data centers. In contrast to such centralized settings, the emerging paradigm of *decentralized training* [53, 39, 24, 22, 21] enables collaborative and democratized machine learning by distributing computation across heterogeneous, geographically dispersed nodes over the Internet, without requiring specialized networking hardware or centralized orchestration.

However, decentralized training presents a core technical challenge: *limited communication bandwidth*. When nodes are connected via commodity networks, communication quickly becomes a bottleneck. Most prior work, has addressed this issue in the context of distributed data parallelism (DDP), where each node maintains a full model replica and synchronizes gradients during training. A variety of bandwidth-efficient techniques, such as gradient quantization [51, 31, 46], sparsification [49, 44, 45], and delayed synchronization [40, 10, 27, 9], have been proposed to reduce overhead in this setting. Pipeline parallelism (PP) [19], where model layers are partitioned across devices, has also been explored to a limited extent [36, 50].

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

A significantly more challenging – and, to our knowledge, entirely unexplored – setting is *context parallelism* (CP) in decentralized environments. CP has become critical for pretraining frontier LLMs, as it enables efficient training with extremely long sequences, often exceeding 100K tokens (e.g., LLAMA 3 [11]: 130K, LLAMA 4: 256K, DEEPSEEK [26]: 128K, QWEN 3 [52]: 128K), thereby enhancing the model's ability to capture long-range dependencies. In context-parallel training, each node processes a local chunk of the input and broadcasts its attention activations to all other nodes in every layer and at every step. This imposes substantial communication demands, as attention mechanisms require *global access* to all key and value activations. While centralized systems handle this using high-bandwidth interconnects, all-to-all communication becomes prohibitively expensive in decentralized environments with low-bandwidth links.

In this work, we propose a method to *drastically reduce* the communication required for context-parallel attention without sacrificing model quality, enabling decentralized systems connected via standard internet-grade links to match the convergence performance of centralized systems with datacenter-grade bandwidth. Our approach leverages the observation that attention activations (queries, keys, and values) often reside on low-dimensional manifolds. We exploit this structure by factorizing the attention weights so that outputs lie within dynamic mixtures of low-dimensional subspaces. To ensure convergence, we optimize the factored weights on a Riemannian product manifold and introduce an efficient reparameterization scheme that significantly reduces computational and communication overhead. Additionally, we provide theoretical guarantees on the expressivity and convergence of our method, offering principled justification for each design choice.

Crucially, this approach introduces only minor architectural changes with negligible training overhead, and these components can be *removed after training*, yielding a standard transformer architecture compatible with existing inference infrastructure and downstream deployment frameworks. We employ our method up to billion-parameter scale models under various settings and demonstrate that our method achieves over $\mathbf{95}\%$ **communication compression** without harming performance, enabling training with long context windows across devices connected via commodity internet ($300\mathrm{Mbps}$), while matching the wall-clock convergence of centralized systems with high-speed interconnects ($100\mathrm{Gbps}$).

2 Background and motivation

2.1 Context-parallel training and the communication bottleneck

We begin with a brief exposition on CP training and refer the reader to [11] for an extended read. Transformer attention requires each query to interact with all key-value pairs, resulting in a computational complexity that grows quadratically with context window. This becomes particularly prohibitive for long sequences, which makes parallelization strategies essential. In *context-parallel* settings, the input sequence $X \in \mathbb{R}^{n\times d}$, where n is the context length and d is the model dimension, is partitioned across m devices along the context dimension:

$$X \;=\; \big[X_1^\top \;\cdots\; X_m^\top\big]^\top, \qquad X_i \in \mathbb{R}^{n_i\times d}, \quad \sum_{i=1}^{m} n_i = n.$$

Each device i computes local queries, keys, and values per head. For clarity we suppress the head index; all quantities are understood to be per attention head unless otherwise stated:

$$Q_i = X_i W_q \in \mathbb{R}^{n_i\times d}, \quad K_i = X_i W_k \in \mathbb{R}^{n_i\times d}, \quad V_i = X_i W_v \in \mathbb{R}^{n_i\times d}.$$

Computing attention locally requires global access to keys and values:

$$K_{\mathrm{g}} = \big[K_1^\top \;\cdots\; K_m^\top\big]^\top \in \mathbb{R}^{n\times d}, \qquad V_{\mathrm{g}} = \big[V_1^\top \;\cdots\; V_m^\top\big]^\top \in \mathbb{R}^{n\times d},$$

Typically, CP performs (some form of) *all-gather*, where each device broadcasts its local K_i, V_i to form $K_{\mathrm{g}}, V_{\mathrm{g}}$, incurring communication cost $O(nd)$ per device where $d \ll n$. Recently proposed *Ring Attention* [29] pipelines this communication in a ring topology, incrementally exchanging local key-value blocks and computing partial attentions at each stage. Above methods fundamentally rely on the costly communication of large K, V matrices.

2.2 Attention Outputs Exhibit Low-Rank Structure

Our compression scheme is inspired by the observation that the attention outputs of pretrained transformers lie on a low-dimensional manifold. To support this, we analyze publicly available checkpoints of large-scale pretrained LLMs and examine their attention activations. Fig. 1 presents an illustration of LLAMA 70B.

Specifically, we measure the *stable rank* of the query (Q), key (K), and value (V) activations across each attention layer. The stable rank of a matrix $A \in \mathbb{R}^{n\times d}$ is defined as: $\operatorname{srank}(A) = \frac{\|A\|_F^2}{\|A\|_2^2}$, where $\|A\|_F$ denotes the Frobenius norm and $\|A\|_2$ the spectral norm. Unlike the conventional matrix rank – which is highly sensitive to small perturbations and numerical noise – the stable rank offers a robust, continuous measure of effective dimensionality. This makes it particularly suitable for characterizing learned neural representations, where numerous singular values are typically small yet non-zero due to noise or over-parameterization.

As depicted in Fig. 1, the stable ranks of attention activations remain low across all layers. Interestingly, Q and K generally exhibit slightly lower ranks than V, indicating a higher degree of compressibility [1]. This observation underpins our approach, leveraging low-rank factorization for efficient compression. Further evidence of this phenomenon in other architectures is provided in Appendix B. Next, we formalize this idea.

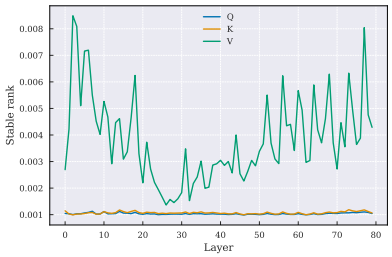

Figure 1: **Attention outputs of LlaMa-70B.** Shown is the empirical rank of the Q, K, and V activations, normalised by their maximum possible rank, for every layer of the official LLAMA 70B checkpoint. All three projections are extremely low-rank: Q and K sit at roughly 0.1% of full rank, while V is slightly larger at $\sim 0.5\%$.

3 Method

We now present our proposed method for efficient context-parallel transformer training. First, we formalize how the empirically observed low-rank structure in attention activations enables effective compression. Next, we explain why using a fixed subspace for compression can be overly restrictive, motivating our joint learning strategy that adaptively optimizes both the projection subspace and the attention weights (§3.1). We then introduce a computationally efficient reparameterization approach that maintains optimality guarantees while significantly reducing overhead (§3.2). Finally, we describe how to reduce communication costs by dynamically compressing attention activations through per-chunk rotations and demonstrate how the model can seamlessly revert to a *standard transformer architecture* at inference time (§3.3–3.5).

In §. 2.2, we saw that the Q, K, V activations of large pretrained transformers exhibit a pronounced low-rank structure (Fig. 1). This finding implies that it is feasible to transmit only the low-dimensional components of these activations between devices, thereby achieving near-lossless compression in practice. Formally, let the columns of an orthonormal matrix $U \in \mathbb{R}^{d\times r}$, with $r \ll d$, span the dominant subspace of the activations. Rather than communicating the full local activation matrix $Z = X^{(i)}W \in \mathbb{R}^{n_i\times d}$, where $Z \in \{K, V\}$ denotes key/value activations, $W \in \{W_k, W_v\}$ and X is the input to the attention layer, we can transmit only its compressed representation: $Z_{\text{comp}} = X^{(i)}WU \in \mathbb{R}^{n_i\times r}$. The original activations can then be reconstructed at the receiving node as: $Z \approx Z_{\text{comp}}U^\top$. This compression method preserves all information within the subspace spanned by U, and is lossless when activations lie entirely in this subspace. Equivalently, this projection can be folded onto the attention weights and be interpreted as factorizing them into a low-rank representation: $W = B(UU^\top), \quad B \in \mathbb{R}^{d\times d}$.

Sub-optimality of a fixed subspace. The formulation above implicitly assumes that an *a priori* choice of U is sufficiently expressive for every layer and every chunk in every optimization stage. It

[1] We only need to compress K and V since Q can remain local.

is straightforward to see where this assumption can break down. Even if there is an optimal low-rank attention weight matrix, restricting weights to the form $W = BUU^\top$ limits the search to the column space of U. If this space does not contain the true optimum, the model may converge to a suboptimal solution. In short, fixing U can prevent the model from reaching the best possible performance.

3.1 Joint Optimization over a Product Manifold

To address the limitations of a fixed U, we propose jointly optimizing factorization $W = B\,U\,U^\top$. Specifically, we simultaneously learn both the subspace representation U and the matrix B on the product manifold: $\mathcal{M} = \mathbb{R}^{d\times d} \times \mathrm{St}(n, r)$. Here, $B \in \mathbb{R}^{d\times d}$ is optimized in standard Euclidean space, whereas U resides on the Stiefel manifold $\mathrm{St}(n, r)$, where updates can be naturally performed via Riemannian gradient descent [2]. The following result establishes that this joint optimization achieves linear convergence under gradient descent.

Convergence. Let $\Phi(W, \vartheta)$ be a smooth loss and consider the factorization $W = B\,U\,U^\top$ for attention weights where $B \in \mathbb{R}^{d\times d},\ U \in \mathrm{St}(d, r),$ and $\vartheta \in \mathbb{R}^p$ denotes all other parameters. Minimizing the reparameterized objective $\hat{\Phi}(B, U, \vartheta) = \Phi\big(BUU^\top, \vartheta\big)$ over the product manifold $\mathcal{M} := \mathbb{R}^{d\times d} \times \mathrm{St}(d, r) \times \mathbb{R}^p$ with Riemannian gradient descent, and under mild assumptions, yields ***Q-linear*** (geometric) convergence to a first-order stationary point. For the formal result and proof, see Lemma 1 (Appendix).

Note that since $\|U\| = 1$, the factorized objective remains Lipschitz smooth, and the convergence result naturally follows from the standard gradient descent theory on both Euclidean and Riemannian manifolds. We include a full proof in Appendix A for completeness, explicitly treating the product manifold structure and assuming a Polyak–Łojasiewicz (PL) condition.

3.2 Reducing Computational Cost via Reparameterization of U

Direct optimisation of U on the Stiefel manifold $\mathrm{St}(n, r)$ via Riemannian gradient descent provides strong theoretical guarantees but is costly: after every Euclidean update, U must be re-orthonormalised (the standard "retraction" on to the manifold), which is performed with a QR or SVD factorisation to restore $U^\top U = I_r$. To mitigate this, we use an efficient reparameterization of U using a fixed orthonormal basis $\overline{U}$ and a learnable rotation $R(\theta) \in O(d)$:

$$U(\theta) \;=\; R(\theta)\,\overline{U},$$

where $O(d)$ denotes the orthogonal manifold consisting of all $d \times d$ orthonormal matrices. If the mapping $\theta \mapsto R(\theta)$ is sufficiently expressive, rotations $R(\theta)$ can fully parameterize $O(d)$, preserving the representational power of the manifold while significantly reducing computational overhead.

Preservation of geometry and stationary points. Reparameterizing U as $U(\theta) = R(\theta)\,\overline{U}$ moves the orthonormal constraint onto an unconstrained Euclidean variable θ, eliminating expensive QR/SVD steps and letting us run ordinary SGD/Adam in θ-space. *A natural concern is that this change of variables might distort the loss landscape and hinder optimization;* however, we show that this is not the case. The chain rule shows $\nabla_\theta \widehat{\Phi}(B, \theta, \vartheta) = D_\theta U(\theta)^\top \,\mathrm{grad}_U \Phi\big(B, U(\theta), \vartheta\big)$, so $\nabla_\theta \widehat{\Phi}$ is exactly the pull-back of the original Riemannian gradient. Thus, the first-order critical points remain unchanged. The following statement formalizes this result.

Equivalence of stationery points. Under the reparameterization $U \;=\; R(\theta)\,\overline{U}$, minimizing $\hat{\Phi}$ possesses exactly the same local minima and strict saddle points as minimizing Φ. For the formal result and proof, see Theorem 1 (Appendix).

Thus, we can effectively represent and optimize the projection subspace implicitly through rotations without compromising the quality or optimality of solutions.

[2] The Stiefel manifold $\mathrm{St}(d, r)$ is defined as the set of all $d \times r$ matrices with orthonormal columns, formally given by $\mathrm{St}(d, r) = \{U \in \mathbb{R}^{d\times r} : U^\top U = I_r\}$.

3.3 Reducing the Communication Cost

The reparameterization $U(\theta) = R(\theta)\,\overline{U}$ allows us to locally cache the fixed orthonormal frame $\overline{U}$ at each node, and transmit only the parameters θ. However, to fully parameterize rotations in the orthogonal group $O(d)$, one typically requires $\frac{1}{2}d(d-1)$ parameters, *i.e.,* $\theta \in \mathbb{R}^{d(d-1)/2}$. We show next that in practice performing a dense search over all possible rotations is unnecessary. Specifically, we can obtain a trade-off between the search space and the communication efficiency by controlling the dimensionality of θ.

To achieve a more compact representation for the communication cost reduction, we select a small set of fixed skew-symmetric matrices $\{A_1, \ldots, A_k\} \subset \mathfrak{o}(d), A_i^T = -A_i$ (where $\mathfrak{o}(d)$ denotes the Lie algebra of the orthogonal group) and define the corresponding k-dimensional Lie subgroup [16, 12]:

$$\mathcal{R}_k = \left\{ R(\theta) = \exp\left(\textstyle\sum_{l=1}^{k} \theta(l) A_i\right) \mid \theta \in \mathbb{R}^k \right\},$$

where $\theta(l)$ is the l^{th} element of θ. Because the exponential map is a local diffeomorphism around $\theta = 0$, the set $U(\theta) = R(\theta)\,\overline{U}$ forms a k-dimensional submanifold of $\mathrm{St}(d, r)$ for sufficiently small $\|\theta\|$. Choosing $k \ll \frac{1}{2}d(d-1)$ thus provides a favorable trade-off between communication cost and representational flexibility. Importantly, our earlier result on the absence of spurious minima remains valid provided an optimal frame $U_\star$ lies within (or sufficiently close to) the reachable manifold $\{R\overline{U} : R \in \mathcal{R}_k\}$, as the mapping $\theta \mapsto U(\theta)$ remains locally surjective onto this manifold.

3.4 Dynamic mixtures of subspaces via per-chunk rotations

§ 3.3 depicted that the rotation dimension k controls a trade-off between representational flexibility and communication efficiency. With a well-chosen a priori $\overline{U}$, it becomes feasible to use a small k, restricting the optimization to a local neighborhood around $\overline{U}$.

We generate this prior through a short, uncompressed **warm-up phase**, in which the model is trained for a small number of iterations (< 500) using a reduced context length to avoid communication bottlenecks. After this phase, each node computes the top r principal components of its local attention weights and stores them as a fixed subspace basis $\overline{U} \in \mathrm{St}(d, r)$. Empirical evidence from prior work on weight–subspace stabilization (e.g., [13, 18]) suggests that dominant activation subspaces emerge early in training, supporting this strategy.

Per-sample adaptation. Using a single global rotation for all inputs may underfit heterogeneous data. To retain expressivity *without increasing* k, we introduce a lightweight mechanism to predict a unique rotation parameter θ for each sequence chunk. Recall that, in context-parallel training, each node i processes a distinct chunk $X_i \in \mathbb{R}^{n_i \times d}$ from the input sequence. For an attention output chunk $Z_i = WX_i$, we employ a small linear prediction head: $\psi : \mathbb{R}^d \to \mathbb{R}^k, \quad \theta = \psi\left(Z_{\text{avg},i}\right)$, where $Z_{\text{avg},i}$ is the average attention output of the chunk, generating chunk-specific rotation parameters. Given a set of preshared skew-symmetric generators $\{A_l\}_{l=1}^k \subset \mathfrak{o}(d)$ cached locally on each node, we construct the rotation as: $R(\theta_i) = \exp\left(\sum_{l=1}^{k} \theta_i(l) A_l\right) \in \mathcal{R}_k \subset O(d)$. Locally, keys and values are compressed as $Z_{\text{comp},i} = Z_i\, R(\theta_i)\,\overline{U} \in \mathbb{R}^{n \times r}$. $(Z_{(\text{comp},i)}, \theta_i)$ is then broadcasted, and the receiving nodes reconstructs the keys/values as: $Z_i \approx Z_{(i,\text{comp})}\,\overline{U}^{\top} R(\theta_i)^{\top}$. Note that peak memory is dominated by the attention computation, scaling as $\mathcal{O}(n_i^2)$, making the linear head's overhead $\mathcal{O}(dk)$ negligible—an observation we also demonstrate empirically. The overall procedure is summarized in Algorithm 1.

Bandwidth cost. In our method, each node transmits nr floats (activations) in $\tilde{Z}$ and k additional scalars in θ. Typically, we have $k \ll nr \ll nd$, ensuring low communication overhead. Remarkably, we found that even using $k = 1$ – a single rotation angle that defines a plane – is sufficient to preserve training stability and input-adaptive flexibility, achieving bandwidth efficiency comparable to that of a fixed global rotation.

In implementation, we set $S \sim \mathcal{N}(0,1)^{d \times d}$ to be fixed and define the skew-symmetric generator $A := \frac{\theta}{\|S - S^\top\|_{\mathrm{F}}}(S - S^\top), \quad A^\top = -A, \quad \theta \geq 0$. For $\theta \in \mathbb{R}$ we set the rotation $R(\theta) = \exp(\theta A) \in O(d)$, so A fixes the rotation plane while θ sets its magnitude.

Algorithm 1 Compression-aware context parallel attention (per node, per head)

Require: Input $X \in \mathbb{R}^{n_i \times d}$, Attention weight $W \in \mathbb{R}^{d \times d}$, Warm-started basis $\bar{U} \in \mathbb{R}^{d \times r}$, learnable linear head $\psi : \mathbb{R}^d \to \mathbb{R}^m$, sync interval c, current step t

Compute local keys and values: $Z \leftarrow XW$
$Z_{\text{avg}} \leftarrow \texttt{MeanToken}(Z)$
$\theta \leftarrow \psi(Z_{\text{avg}})$ ▷ Compute rotation param from local chunk
$U \leftarrow R(\theta)\,\bar{U}$ ▷ Construct data-dependent subspace
Compress: $Z_{\text{comp}} \leftarrow ZU$
Broadcast $(Z_{\text{comp}}, \theta)$ to all other nodes
Receive $(Z_{(\text{comp},j)}, \theta_j)$ from all other nodes j
for all received $(Z_{(\text{comp},j)}, \theta_j)$ **do**
    $U_j \leftarrow R(\theta_j)\,\bar{U}$
    $Z_j \leftarrow Z_{(\text{comp},j)} U_j^\top$ ▷ Decompress
end for
Aggregate global $Z_g \in \{K_j, V_j\}, \forall j$ from all nodes
Compute blockwise attention: $A \leftarrow \texttt{Softmax}(QK^\top/\sqrt{d})V$
if $t \bmod c = 0$ **then**
    $W \leftarrow \texttt{AllReduceAvg}(W)$ ▷ Sparse sync of attention weights
end if

Second-order approximation. Since A is skew-symmetric, its spectral norm satisfies $\|A\|_2 = \theta$. For sufficiently small $|\theta| \leq \epsilon \ll 1$, the rotation matrix $R(\theta)$ admits a second-order Taylor approximation:

$$R(\theta) \approx I + \theta A + \tfrac{1}{2}\theta^2 A^2. \tag{1}$$

This approximation provides two key advantages. 1) **Computational cost:** scaling as $\mathcal{O}(d^2)$, in contrast to the exact matrix exponential computation (e.g., by Padé or Schur decomposition), which scales as $\mathcal{O}(d^3)$. 2) **Near identity bias:** it induces a beneficial near-identity bias, effectively acting as an approximately unbiased estimator of identity I when θ is small and centered around zero (enforced via clipping). In this regime, higher-order terms vanish in expectation, yielding $\mathbb{E}[R(\theta)] \approx I$. This property allows rotations to remain close to the initial warm-start subspace $\overline{U}$, facilitating controlled local adaptation without significant drift. By fixing A and using a scalar θ, we achieve a communication complexity of $\mathcal{O}(nr)$, significantly lower than the naive $\mathcal{O}(nd)$.

Attention weights must still be synchronized across devices, but they evolve far more slowly than activations [6, 5]. We therefore average the corresponding weights only every c steps; in all experiments we use $c = 200$, which incurs negligible communication overhead.

3.5 Unplugging the Projection Components

Our method augments the transformer architecture with two non-standard components: (i) a small linear *rotation head* predicting the rotation parameters θ, and (ii) low-rank *projection matrices* U used for compressing activations. Although these components pose minimal overhead during training, strict API compatibility with off-the-shelf transformer models might be necessary for certain downstream applications.

As training proceeds, the learnable weights associated with our auxiliary projection heads collapse onto the data–dependent subspaces they steer. Once the model is close to convergence we can therefore *drop these heads entirely*, reverting to a vanilla Transformer without losing the predictive gains accumulated during training. The following result formalizes the collapse mechanism.

Bound on "idle" attention directions with data dependent projectors. Let the sample projector be $P(x) = U(x)U(x)^\top$. Pick any other projector Q that projects onto an arbitrary subspace. Define the average overlap $p_Q := \mathbb{E}_x\big[\|P(x)Q\|_2\big] \in [0, 1]$. Run stochastic gradient descent with weight decay $\lambda > 0$. Then, the attention weights that lie inside the Q-subspace obey $\lim_{t\to\infty} \big\|W^{(t)}Q\big\|_F \leq \frac{p_Q\, L}{\lambda}$ for an L Lipshchitz bounded loss. Hence, if the data almost never excites those directions ($p_Q \ll 1$), the corresponding weights shrink away. That is, idle directions are pruned for free. For the formal theorem and proof, see Theorem 3 (Appendix).

Once the weights have collapsed onto their data–aligned subspaces, both the rotation head and its basis matrix U are redundant. We can therefore *detach* these components and perform a brief,

low-learning-rate fine-tuning pass to polish the remaining parameters. As Fig. 4 shows, the loss curve remains smooth across this transition, indicating that no *optimisation shock* is introduced.

At inference time the model is now *indistinguishable* from a standard transformer: it adds **zero** extra parameters, requires no custom kernels, and is fully compatible with existing deployment pipelines.

4 Related Work

Decentralized training. Decentralized learning dispenses with a central coordinator, instead relying on a collective of *autonomous* devices that cooperate over mesh-style networks to train large-scale models. These devices are typically heterogeneous and geographically dispersed, confronting links of nonuniform latency and bandwidth. The foundational theory on convergence and robustness has been established by [24, 22, 21], while complementary systems work has demonstrated practical viability on real clusters [41, 7]. Most prior art, however, is confined to DDP settings [24, 22, 21, 7], limiting model size to the aggregate memory of an individual node. Note that this is a comparatively well studied domain, and is orthogonal to the unexplored decentralized context parallel setting that we explore. A notable work in DDP domain is Power Gossip [48], which replaces synchronous all-to-all communication with gossip-style information exchange among neighboring replicas arranged in a mesh. Its key insight is that, when each replica trains independently via local SGD, the pairwise weight differences evolve in a low-rank sub-space, enabling them to efficiently compress the weight differences during gossip. Another interesting DDP method is Photon [42], where its communication savings stem primarily from infrequent gradient exchanges rather than from any explicit compression scheme. Such skip-sync approaches are infeasible in context-parallel pipelines, where activations must be transferred between nodes at every forward and backward pass. Nevertheless, these DDP-style techniques are orthogonal to our method and could be combined with it in hybrid setups.

Scheduling-oriented approaches such as SWARM parallelism [39] and Tasklets [53] alleviate straggler effects and network stochasticity, yet they still inherit the communication overhead intrinsic to the decentralized setting. In contrast, we introduce the first communication-compression strategy tailored to CP, removing a critical bandwidth bottleneck that has thus far hindered scaling decentralized models across larger context windows.

Context-parallel attention. For single-device long-sequence processing, sparse approximations such as BigBird [54] cut attention complexity to $O(n)$, while IO-aware exact kernels like FlashAttention [33] maximize hardware throughput with tiling and on-chip caching. Recent systems research parallelizes the *sequence dimension* itself [25, 37, 15]: Blockwise Parallel Transformers overlap compute and ring-all-reduce to achieve near-linear speedups on sequences of 32K tokens [28], and RingAttention extends the idea to virtually unlimited contexts via pipelined block exchanges [29]. These methods, however, still broadcast full key/value tensors. Our approach instead transmits a compact low-rank representation plus a lightweight rotation, reducing bandwidth while preserving exact attention semantics and thus complementing existing context-parallel frameworks.

5 Experiments

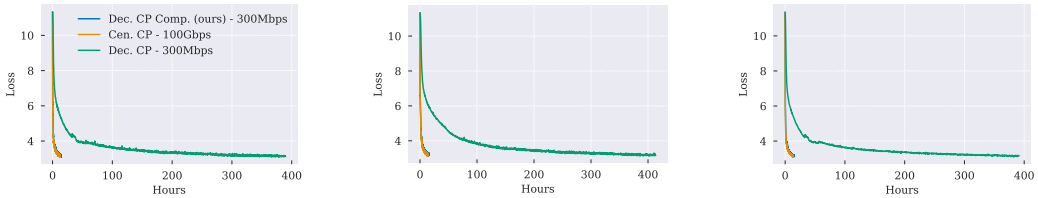

Figure 2: **Convergence in low-bandwidth settings.** From left to right: Fineweb, C4, and BookCorpus. The training curves are presented against wall-clock time for an 8-layer (800M) model trained with a 132K context window parallelized across 8 GPUs. Decentralized models utilize 300Mbps connections while the centralized model has datacenter-grade 100Gbps links. Our compressed model achieves on-par convergence to the centralized model, even under a 300Mbps bandwidth budget. In contrast, the non-compressed decentralized model with 300Mbps links suffers from significantly slower convergence.

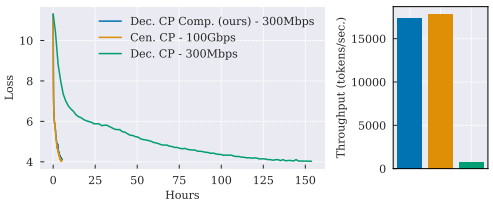

Figure 3: **Scaling across parallelism strategies.** Our compression based CP scheme can be seamlessly fused with other parallel training strategies. We train a 3B-parameter model (32 layers) with both pipeline parallel and CP enabled across 32 A100s. Our compressed approach yields substantial throughput gains over uncompressed decentralized CP and nearly matches the performance of centralized CP.

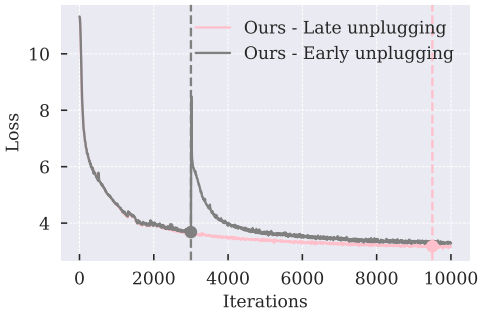

Figure 4: **Unplugging projection components.** After sufficient training, the rotation head and projection layers can be removed—reverting the network to a vanilla transformer—without impairing convergence. Training curves with dashed lines marking projection removal points. Late removal preserves convergence (see §. 3.5), while early removal causes a temporary disruption followed by surprisingly rapid recovery.

5.1 Experimental Setup

We evaluate decoder-only models on three large-scale corpora – FineWeb (FW) [30], C4 [35], and BookCorpus (BC) [56]. For each dataset, we reserve $10\,\%$ of the training split for validation. All model backbones follow LLAMA 3 [11]; exact model specifications are given in the corresponding sections. We use a `base-learning-rate` = 3×10^{-4} with linear warm-up and decay, and apply a `weight-decay` = 0.01. Every transformer layer is compressed except for the final block, where K and V projections are compressed by $98\,\%$ and $95\,\%$ (overall 96.5%), respectively by choosing r w.r.t. d appropriately. We use the GPT2 tokenizer for all models.

5.2 Bandwidth efficiency in decentralized settings

We train an 8-layer, 800M-parameter model (`embedding-size` = 2048, `attention-heads` = 8) under two network settings: a centralized 100Gbps fabric and decentralized 300Mbps internet-grade links. Using CP, we process a sequence length of 132K tokens across eight A100 GPUs connected at the respective bandwidths. Fig. 2 shows that vanilla CP over a 300Mbps link is more than $20\times$ slower compared to a centralized 100Gbps mesh. With our compression, the same 300Mbps setup converges almost as fast as the centralized baseline.

Table 1: **Design ablations** (val. perplexity ↓). All models are trained for 10K steps with a 132K context. Second-order approximations preserve performance, while overcompressing V degrades it.

SETTING	FW	C4	BC
Ours	**22.64**	**23.33**	**25.27**
Ours + Fixed $\bar{U}$	26.57	27.11	30.33
Ours + Rand. rot. $R(\theta)$	24.93	25.17	29.58
Ours - 2nd-order approx.	**22.64**	**23.33**	**25.27**
Ours - No warm start	26.63	26.91	30.15
Ours ($K, V \to 98\%$)	24.74	24.99	29.46
Ours ($K \to 99\%, V \to 95\%$)	24.68	24.91	29.22

Validation. Table 2 reports test-time performance of the trained models. To this end, we train each model up to its compute-optimal point, following the Chinchilla scaling law [17]. Specifically, for our 800M-parameter models, we use a $1:20$ model-to-token ratio and train for 16B tokens on each dataset. Remarkably, our compressed decentralized model matches, and even slightly outperforms, the perplexity of the centralized model at the same number of training iterations, while delivering significantly higher throughput than vanilla (uncompressed) CP over commodity links. Training the uncompressed model to completion over low-bandwidth links is computationally infeasible (estimated at over 150 days), so we report only its throughput (TPS) in this setting.

Table 2: **Validation perplexity (↓) and throughput (TPS).** All models are trained with a 132K context window to the compute-optimal point [17] (16B training tokens). Our method yields a 20× TPS boost while slightly outperforming centralized CP in perplexity, with minimal memory overhead.

MODEL	FW	C4	BC	TPS	MEM (GB)/GPU
Cen. CP - 100Gbps	17.18	17.51	17.88	56K	38.4
Dec. CP - 300Mbps†	–	–	–	2.7K	38.4
Dec. CP Comp. - 300Mbps (ours)	**17.06**	**17.47**	**17.81**	**55K** (×20)	38.7 (+0.7%)

† Training uncompressed models to convergence at 300Mbps is infeasible (>150 days); only throughput is reported.

5.3 Ablations

We perform ablations on 800M-parameter models with a 132K context across eight A100 GPUs (see Table 1). Models using learned rotations outperform those with fixed or random projections. The second-order exponential approximation does not impact performance, confirming its adequacy. Omitting the warm-start initialization of principal directions ($\bar{U}$) noticeably degrades results, highlighting the importance of this prior.

Scaling: Our compression based CP scales well and can be seamlessly fusing with other parallel training strategies. We scale the model to 32 layers (3B parameters) with both pipeline parallel and CP enabled over 32 A100s (Fig. 3) and achieve a significant throughput gain.

Reparameterization: A key step of our method is reparameterizing U which bypasses expensive Riemannian operations (QR/SVD pullbacks). As shown in Table 4, this reparameterization significantly improves throughput (TPS). More ablations against architecture choices are provided in Appendix C.

Table 3: **Effect of warmup steps** (val. perplexity ↓). All models are trained for 10K steps with a 132K context. The method is not highly sensitive to the number of warmup steps.

WARMUP-STEPS	PERPLEXITY
0	26.63
100	24.44
300	22.66
500	22.64
1000	22.87
2000	22.64
5000	22.71

Warmup steps: To measure the effect warmup steps of we conducted an ablation study varying the warm-up duration and evaluated the resulting perplexity on the FineWeb dataset. The results are shown in Table 3. As demonstrated, even with a reduced warm-up of 300 steps, the model achieves comparable performance, indicating no significant degradation. In practice, we default to 500 steps to provide a safe and stable baseline. This study further emphasizes the lightweight and robust nature of our warm-up strategy, especially in contrast to the more elaborate scheduling mechanisms commonly employed in modern LLM pre-training. Note that the perplexity differences are minor and stable, indicating performance is stable after 300 warm-up steps.

5.4 Unplugging Projections and Rotation Heads

As discussed in §. 3.5, practitioners may prefer reverting to a standard transformer after pretraining for compatibility with downstream frameworks. We empirically validate our theoretical prediction that attention weights progressively align with the projection subspace, allowing safe removal of projection layers and rotation heads near the end of training. Fig. 4 shows that removing these components late preserves convergence, while doing so prematurely disrupts training.

5.5 Comparison Against Baselines

As no prior baselines exist for CP compression, we construct two: (i) **Sparsification**—a Top-10% scheme (90% compression), transmitting only the largest-magnitude entries of the K, V chunks, inspired by common DDP compression methods; (ii) **Quantization**—a 4-bit quantization (75% compression) of the K, V activations prior to transmission, following standard practices in activation

compression. As shown in Fig. 5 (left), we outperform these baselines comprehensively (132k context window) even when using a more aggressive compression rate of 96.5%.

For completeness, we also compare against long-context models BigBird [54] and CosFormer [34], which are not designed for CP and can handle at most 32K tokens on an A100. For a fair comparison, we apply our compression to CP across four GPUs, each processing 8K tokens. As shown in Fig. 5 (right), both baselines exhibit significantly worse convergence than our method. All experiments are performed on 800M parameter models.

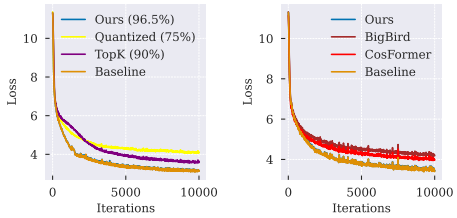

Figure 5: **Baseline comparisons.** *Left:* Because no method currently compresses context-parallel training, we build two baselines; Top-k sparsification and quantization. *Right:* We also compare with long-context models BigBird and CosFormer. Both are limited to 32K tokens on A100 GPUs, so all models are evaluated at that length. In both panels, our compressed CP curve is nearly indistinguishable from the uncompressed reference, whereas every baseline falls well short.

6 Conclusion

We propose the first compression method that enables context-parallel training of language models in decentralized environments with low-bandwidth interconnects. Our approach supports training with context lengths over 100K tokens on isolated GPUs connected via internet-grade links (e.g., 300Mbps), while matching the wall-clock convergence of centralized systems with high-speed (100Gbps) connections. Additionally, our method preserves compatibility with standard transformer architectures by allowing the projection layers to be removed after training, facilitating seamless deployment in downstream frameworks. We provide a theoretical analysis of the key properties of our method and validate its effectiveness through an extensive empirical evaluation.

7 Limitations

Our compression method delivers near-lossless convergence in context-parallel training, but several open questions remain. First, alternative reparameterisations beyond simple subspace rotations may unlock further accuracy or efficiency gains. Second, the method's surprising ability to locate good minima even as the search space is heavily reduced (via very low-dimensional θ) lacks a rigorous explanation; its ties to recent work on implicit regularisation and lottery-ticket-style phenomena deserve closer study. Despite these gaps, this work establishes the first baseline for context-parallel compression and we hope it spurs deeper theoretical and empirical exploration.

Table 4: **TPS gain from design choices.** Reparameterization and second-order approximation yield significant throughput improvements.

SETTING	TPS (↑)
Ours	**55K**
w/o reparam.	37K
w/o 2^{nd} ord. approx.	30K

References

[1] Emmanuel Abbe, Samy Bengio, Enric Boix-Adsera, Etai Littwin, and Joshua Susskind. Transformers learn through gradual rank increase. *Advances in Neural Information Processing Systems*, 36, 2024.

[2] P.-A. Absil, Robert Mahony, and Rodolphe Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, 2008.

[3] Alibaba DAMO Academy. Qwen: Efficient and scalable language models. *arXiv preprint arXiv:2309.16609*, 2023.

[4] Allen AI. Olmo: Open language models. *arXiv preprint arXiv:2402.02309*, 2024.

[5] Jimmy Ba, Geoffrey Hinton, Volodymyr Mnih, Joel Z. Leibo, and Catalin Ionescu. Using fast weights to attend to the recent past. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

[6] Chen Chen, Hong Xu, Wei Wang, Baochun Li, Bo Li, Li Chen, and Gong Zhang. Synchronize only the immature parameters: Communication-efficient federated learning by freezing parameters adaptively. *IEEE Transactions on Parallel and Distributed Systems*, 2023. Early access.

[7] Michael Diskin, Alexey Bukhtiyarov, Max Ryabinin, Lucile Saulnier, Anton Sinitsin, Dmitry Popov, Dmitry V Pyrkin, Maxim Kashirin, Alexander Borzunov, Albert Villanova del Moral, et al. Distributed deep learning in open collaborations. *Advances in Neural Information Processing Systems*, 34:7879–7897, 2021.

[8] Yihe Dong, Jean-Baptiste Cordonnier, and Andreas Loukas. Attention is not all you need: Pure attention loses rank doubly exponentially with depth. In *International Conference on Machine Learning*, pages 2793–2803. PMLR, 2021.

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

[10] 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.

[11] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[12] Alan Edelman, Tomás A. Arias, and Steven T. Smith. The geometry of algorithms with orthogonality constraints. *SIAM Journal on Matrix Analysis and Applications*, 20(2):303–353, 1998.

[13] Ruili Feng, Kecheng Zheng, Yukun Huang, Deli Zhao, Michael Jordan, and Zheng-Jun Zha. Rank diminishing in deep neural networks. *Advances in Neural Information Processing Systems*, 35:33054–33065, 2022.

[14] Elias Frantar and Dan Alistarh. Gptq: Accurate post-training quantization for generative pretrained transformers. *arXiv preprint arXiv:2303.00775*, 2023.

[15] 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.

[16] Brian C. Hall. *Lie Groups, Lie Algebras, and Representations: An Elementary Introduction*, volume 222 of *Graduate Texts in Mathematics*. Springer, Cham, 2 edition, 2015.

[17] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models.

[18] Yuxuan Hu, Jing Zhang, Zhe Zhao, Chen Zhao, Xiaodong Chen, Cuiping Li, and Hong Chen. Sp^3: Enhancing structured pruning via pca projection. In *ACL (Findings)*, 2024.

[19] Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Dehao Chen, Mia Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V Le, Yonghui Wu, et al. Gpipe: Efficient training of giant neural networks using pipeline parallelism. *Advances in neural information processing systems*, 32, 2019.

[20] Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Big transfer (bit): General visual representation learning. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part V 16*, pages 491–507. Springer, 2020.

[21] Anastasia Koloskova, Nicolas Loizou, Sadra Boreiri, Martin Jaggi, and Sebastian Stich. A unified theory of decentralized sgd with changing topology and local updates. In *International Conference on Machine Learning*, pages 5381–5393. PMLR, 2020.

[22] Anastasia Koloskova, Sebastian Stich, and Martin Jaggi. Decentralized stochastic optimization and gossip algorithms with compressed communication. In *International Conference on Machine Learning*, pages 3478–3487. PMLR, 2019.

[23] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

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

[25] Wanchao Liang, Tianyu Liu, Less Wright, Will Constable, Andrew Gu, Chien-Chin Huang, Iris Zhang, Wei Feng, Howard Huang, Junjie Wang, et al. Torchtitan: One-stop pytorch native solution for production ready llm pre-training. *arXiv preprint arXiv:2410.06511*, 2024.

[26] Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

[27] Bo Liu, Rachita Chhaparia, Arthur Douillard, Satyen Kale, Andrei A Rusu, Jiajun Shen, Arthur Szlam, and Marc'Aurelio Ranzato. Asynchronous local-sgd training for language modeling. *arXiv preprint arXiv:2401.09135*, 2024.

[28] Hao Liu and Pieter Abbeel. Blockwise parallel transformers for large context models. *Advances in neural information processing systems*, 36:8828–8844, 2023.

[29] Hao Liu, Matei Zaharia, and Pieter Abbeel. Ring attention with blockwise transformers for near-infinite context. In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2310.01889.

[30] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models, 2016.

[31] Roula Nassif, Stefan Vlaski, Marco Carpentiero, Vincenzo Matta, Marc Antonini, and Ali H Sayed. Quantization for decentralized learning under subspace constraints. *IEEE Transactions on Signal Processing*, 71:2320–2335, 2023.

[32] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, 2nd edition, 2006.

[33] Matteo Pagliardini, Daniele Paliotta, Martin Jaggi, and François Fleuret. Faster causal attention over large sequences through sparse flash attention. *arXiv preprint arXiv:2306.01160*, 2023.

[34] Zhen Qin, Weixuan Sun, Hui Deng, Dongxu Li, Yunshen Wei, Baohong Lv, Junjie Yan, Lingpeng Kong, and Yiran Zhong. cosformer: Rethinking softmax in attention. *arXiv preprint arXiv:2202.08791*, 2022.

[35] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv e-prints*, 2019.

[36] Sameera Ramasinghe, Thalaiyasingam Ajanthan, Gil Avraham, Yan Zuo, and Alexander Long. Beyond top-k: Structured sparsification for compression in pipeline parallel. In *ICLR 2025 Workshop on Modularity for Collaborative, Decentralized, and Continual Deep Learning*.

[37] Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 3505–3506, 2020.

[38] Xiaozhe Ren, Pingyi Zhou, Xinfan Meng, Xinjing Huang, Yadao Wang, Weichao Wang, Pengfei Li, Xiaoda Zhang, Alexander Podolskiy, Grigory Arshinov, et al. Pangu-{\Sigma}: Towards trillion parameter language model with sparse heterogeneous computing. *arXiv preprint arXiv:2303.10845*, 2023.

[39] Max Ryabinin, Tim Dettmers, Michael Diskin, and Alexander Borzunov. Swarm parallelism: Training large models can be surprisingly communication-efficient. In *International Conference on Machine Learning*, pages 29416–29440. PMLR, 2023.

[40] Max Ryabinin, Eduard Gorbunov, Vsevolod Plokhotnyuk, and Gennady Pekhimenko. Moshpit sgd: Communication-efficient decentralized training on heterogeneous unreliable devices. *Advances in Neural Information Processing Systems*, 34:18195–18211, 2021.

[41] Max Ryabinin and Anton Gusev. Towards crowdsourced training of large neural networks using decentralized mixture-of-experts. *Advances in Neural Information Processing Systems*, 33:3659–3672, 2020.

[42] Lorenzo Sani, Alex Iacob, Zeyu Cao, Royson Lee, Bill Marino, Yan Gao, Dongqi Cai, Zexi Li, Wanru Zhao, Xinchi Qiu, et al. Photon: Federated llm pre-training. *arXiv preprint arXiv:2411.02908*, 2024.

[43] Shuvam Sanyal, Angela Fan, and Eric Wallace. Inheritune: Leveraging inherent low-rank structure in transformers. *arXiv preprint arXiv:2404.03789*, 2024.

[44] Shaohuai Shi, Qiang Wang, Kaiyong Zhao, Zhenheng Tang, Yuxin Wang, Xiang Huang, and Xiaowen Chu. A distributed synchronous sgd algorithm with global top-k sparsification for low bandwidth networks. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pages 2238–2247. IEEE, 2019.

[45] Shruti Singh and Shantanu Kumar. Efficient distributed training through gradient compression with sparsification and quantization techniques. *arXiv preprint arXiv:2502.07634*, 2024.

[46] Hanlin Tang, Ce Zhang, Shaoduo Gan, Tong Zhang, and Ji Liu. Decentralization meets quantization. *arXiv preprint arXiv:1803.06443*, 3, 2018.

[47] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[48] Thijs Vogels, Sai Praneeth Karimireddy, and Martin Jaggi. Powergossip: Practical low-rank communication compression in decentralized deep learning. *arXiv preprint arXiv:2008.01425*, 2020.

[49] Haozhao Wang, Song Guo, Zhihao Qu, Ruixuan Li, and Ziming Liu. Error-compensated sparsification for communication-efficient decentralized training in edge environment. *IEEE Transactions on Parallel and Distributed Systems*, 33(1):14–25, 2021.

[50] Hongyi Wang, Saurabh Agarwal, and Dimitris Papailiopoulos. Pufferfish: Communication-efficient models at no extra cost. *Proceedings of Machine Learning and Systems*, 3:365–386, 2021.

[51] Jiaxiang Wu, Weidong Huang, Junzhou Huang, and Tong Zhang. Error compensated quantized sgd and its applications to large-scale distributed optimization. In *International conference on machine learning*, pages 5325–5333. PMLR, 2018.

[52] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

[53] Binhang Yuan, Yongjun He, Jared Davis, Tianyi Zhang, Tri Dao, Beidi Chen, Percy S Liang, Christopher Re, and Ce Zhang. Decentralized training of foundation models in heterogeneous environments. *Advances in Neural Information Processing Systems*, 35:25464–25477, 2022.

[54] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, 33:17283–17297, 2020.

[55] Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. Galore: Memory-efficient llm training by gradient low-rank projection. *arXiv preprint arXiv:2403.03507*, 2024.

[56] Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Section 1

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 7

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: Appendix

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 5

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Provided in supplementary materials

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 5 and Appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Experiments are too expensive and therefore it is computationally expensive to report error bars.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Section 5

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Implications are discussed in Section 1

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate

to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.

- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: No data or models that are of high rist of misuse.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Section 5

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [No]

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: No human participants were used in the study.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: No study participants were used in the sudy.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [No]

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.