Learned Subspace Compression for Communication-Efficient Pipeline Parallelism

Anonymous Authors[1]

Abstract

Pipeline parallelism enables training of large language models that exceed single-device memory, yet inter-stage activation communication becomes the dominant bottleneck when training spans low-bandwidth, heterogeneous networks. Recent work in this area (Ramasinghe et al., 2025) has proposed using fixed orthogonal projections to compress activations. However, this still results in a significant performance degradation and requires intrusive training changes for constrained optimization. A natural alternative is to learn a low rank projection for each pipeline stage, however maintaining the necessary orthogonality of these projectors during training remains a challenge. We present **MDCP-PP** (Manifold and Dictionary Constrained Projection for Pipelined Parallelism), a method that treats inter-stage compression as a learnable orthogonal projection under explicit Stiefel manifold (orthogonal matrices) constraints. Rather than prescribing a fixed global subspace, MDCP-PP lets each pipeline stage discover and continuously adapt its own task-optimal compression subspace via manifold-constrained steepest descent (Yang & Lai, 2026). To recover token-specific signals at stage boundaries, we introduce per-stage factorized anchor embeddings that allow for full-rank activation reconstruction with negligible communication overhead. We further incorporate residual vector quantization with a streaming codebook synchronization protocol that amortizes dictionary communication. Across LLaMA (Touvron et al., 2023a) models from 150M to 1B parameters and 4 to 8 stage pipelines, MDCP-PP achieves 4×–8× inter-stage compression within ∼2% of the uncompressed validation loss, and extends to 16× compression with vector quantization at ∼3% degradation.[1]

1. Introduction

Training large-scale foundation models across geographically distributed, heterogeneous hardware introduces communication challenges that centralized distributed training systems (Rajbhandari et al., 2020) were never designed to handle (Douillard et al., 2023; Jaghouar et al., 2024; Ramasinghe et al., 2025; Sarfi et al., 2025; Lidin et al., 2026; Nabli et al., 2025). Centralized systems assume tightly coupled accelerator clusters with high-bandwidth interconnects, but real-world decentralized deployments often rely on consumer-grade GPUs as their backbone and must operate over commodity wide-area networks with limited bandwidth. Since, modern models have grown well beyond the memory capacity of these individual accelerators, model parallelism is a practical necessity. Pipeline parallelism (Huang et al., 2019) responds by partitioning parameters across devices, allowing GPUs to easily host a few layers each. However, this introduces a new bottleneck that every micro-batch triggers activation exchanges across stages in both the forward and backward passes, saturating the network bandwidth.

This communication cost motivates *low-bandwidth pipeline-parallel training*, in which intermediate activations are aggressively compressed before transmission (Ramasinghe et al., 2025). Beyond accessibility, compressing inter-stage activations is empirically well-motivated. There is growing evidence that foundation models converge to intrinsically low-rank solutions (Janson et al., 2026; Ramasinghe et al., 2025; Yang et al., 2023; Galanti et al., 2025), suggesting that the full-rank activation tensors exchanged between pipeline stages carry significant redundancy. Exploiting this structure allows us to eliminate communication overhead and transmit only the information that actually drives learning.

This setting is fundamentally harder than compression in data-parallel training (Sarfi et al., 2025; Vogels et al., 2019a), where compressed quantities are gradients computed independently across identical model replicas. In data-parallel

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

[1]Code is available at https://anonymous.4open.science/r/learned-pipeline-2616/

systems, compression errors are naturally averaged out among workers and remain largely decoupled from the forward computation. Activation compression operates under an entirely different regime: because pipeline stages hold complementary, non-overlapping subsets of the model, so the compressed representation produced by one stage serves as the direct input to the next. Any distortion introduced at transmission does not average away. Instead, it propagates forward through subsequent layers and backward through the gradient computation, accumulating across stages and corrupting the exact signal that drives learning. Consequently, even minor compression artifacts can have immediate and compounding consequences for final model quality.

Recent work on Subspace Networks (SSNs) (Ramasinghe et al., 2025) addresses this by constraining each activation of width d to a fixed, shared low-rank subspace of rank r defined by an orthogonal matrix $U_r \in \mathbb{R}^{d\times r}$. Specifically, they transmit activations $h \in \mathbb{R}^d$ as low-dimensional coordinates $z = U_r^\top h$ and reconstruct them locally as $\hat{h} = U_r z$. While SSNs confirm that architectural modification is a viable design space for communication-efficient training, the approach requires intrusive constrained optimization: model weights are restricted to a common low-rank subspace, a modified AdamW (Kingma & Ba, 2015; Loshchilov & Hutter, 2019) optimizer is needed to maintain a weight in the subspace, and a static embedding offset is applied at each stage (See (§B)). Consequently, we observe substantial accuracy degradation under a token-matched setting relative to uncompressed baselines.

Rather than prescribing a global basis for all layers, we let each pipeline stage *learn* its own communication subspace jointly with the task objective. Intermediate activations exhibit substantial low-rank structure even under modern optimizers (Jordan et al., 2024; Liu et al., 2025), suggesting that communication-efficient representations can naturally emerge end-to-end. However naively learning these projectors degrades the performance because standard gradient updates carry projectors off the Stiefel manifold (the set of all matrices with orthonormal columns), destroying the orthonormality on which isometric compression depends. Once a projector escapes the manifold, the model begins encoding features outside the intended subspace, leading to severe performance degradation. Crucially, we observe that these unconstrained models underperform even fixed orthogonal baselines, confirming that naive learning without manifold awareness is strictly worse than no learning at all (see §D). We identify this manifold escape as the principal failure mode and address it using manifold-constrained steepest-descent updates (Yang & Lai, 2026), which keep projectors strictly on the Stiefel manifold throughout training. Furthermore, removing the global subspace eliminates the need for SSN-style embedding decompositions. Instead, we introduce *factorized anchor embeddings* (Lan et al., 2020), a low-rank factorization $E = E_p^{\text{small}} P_p$ with fully trainable matrix $E_p^{\text{small}} \in \mathbb{R}^{V\times r}$ and a frozen matrix $P_p \in \mathbb{R}^{r\times d}$, where V is the vocabulary size. This approach keeps the parameter count low while permitting the effective embedding to recover its full rank at each stage. Finally, we further reduce bandwidth via vector quantization (VQ) on the low-rank manifold, employing a lightweight dictionary-stream protocol that amortizes codebook synchronization across many activation exchanges.

To this end, we introduce **Manifold and Dictionary Constrained Projection for Pipeline Parallelism (MDCP-PP)**, a method that treats inter-stage communication as a learnable geometric projection rather than a fixed architectural constraint. Our main contributions are:

- **Learned per-stage communication subspaces.** We demonstrate that enforcing a shared global subspace across pipeline stages actively harms learning in token-matched settings. Instead, allowing each stage to dynamically learn its own compression subspace on the Stiefel manifold offers a more natural and effective approach to low-bandwidth activation transmission (§ 3.2).
- **Factorized anchor embeddings and vector quantization.** We introduce a low-rank embedding factorization that eliminates the need for intrusive SSN-style token embedding decompositions, allowing the network to transmit only integer token IDs at a negligible cost. We pair this with vector quantization and a streaming codebook synchronization protocol to further minimize inter-stage bandwidth. (§ 3.5)
- **Strong empirical performance across model scales.** We evaluate MDCP-PP on models ranging from 150M to 1B parameters. We demonstrate that our method recovers downstream performance to within 1% of the uncompressed baseline in token-matched settings, consistently outperforming SSNs by a 5% margin across all evaluated scales (§ 4.2).

2. Related Works

Low Bandwidth Pre-training. The growing utility of large AI systems has motivated efforts to democratize large-scale model training across decentralized bodies of participants connected over bandwidth-constrained networks such as the open internet. Early works on decentralized optimization established theoretical foundations: (Lian et al., 2017) showed that decentralized SGD can match centralized convergence rates, while subsequent analyses unified gossip-based optimization with compressed communication and local updates (Koloskova et al., 2019; 2020). To make

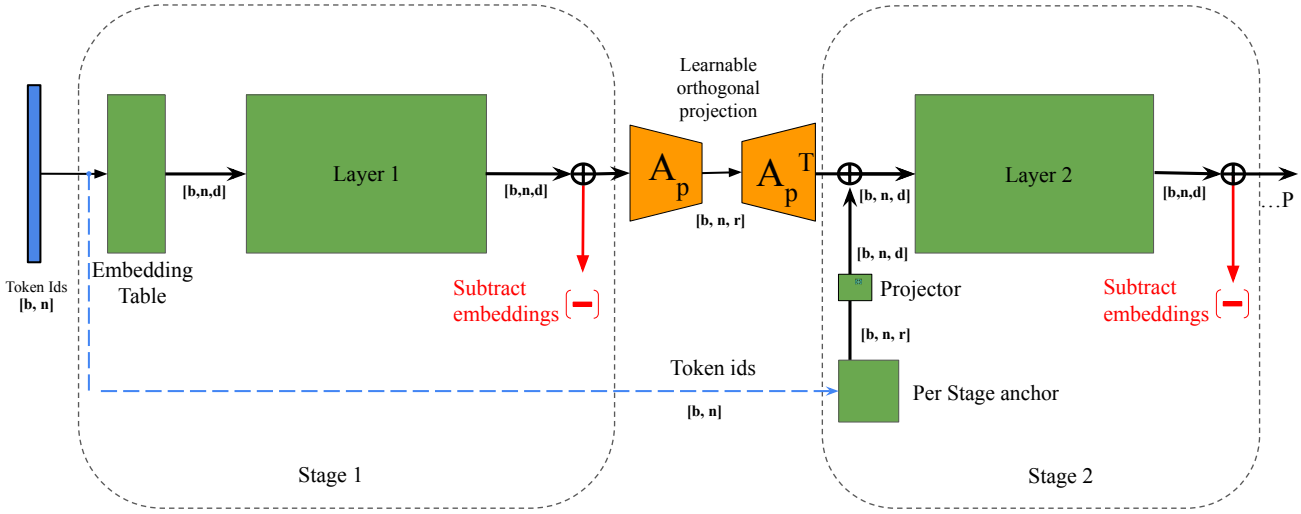

Figure 1. **Overview of MDCP-PP compression at a pipeline stage boundary, repeated across all $P-1$ inter-stage boundaries.** At each boundary, the token-dependent offset is subtracted from the boundary activation $X^{b_p} \in \mathbb{R}^{b\times n\times d}$ before transmission (red arrow): the first stage subtracts the original token embeddings, while subsequent stages subtract their per-stage factorized anchor embeddings $E_p^{\text{small}}[t_{ids}]\,P_p$, where P_p is a frozen orthogonal projector. The residual is then projected to a low-dimensional representation $Z^{b_p} \in \mathbb{R}^{b\times n\times r}$ via a learnable orthogonal projector $A_p \in \mathrm{St}(d,r)$, reducing inter-stage communication volume by a factor of d/r. At the receiving stage, $A_p^\top$ reconstructs the full-dimensional activation, to which the destination-stage factorized anchor is added to restore the offset. Integer token IDs are transmitted alongside the compressed activation on the same channel at negligible cost (blue dashed line), enabling each stage to look up its local anchor without additional bandwidth overhead.

distributed training communication-efficient, a rich line of work explored gradient compression through sparsification (Wang et al., 2017; Wangni et al., 2018; Lin et al., 2018), scalar and sign-based quantization (Alistarh et al., 2017; Bernstein et al., 2018; Tang et al., 2021; Dettmers et al., 2022), low-rank approximation (Vogels et al., 2019b), and error-feedback mechanisms that correct biased compressors (Wu et al., 2018; Karimireddy et al., 2019). Building on these foundations, volunteer-style collaborative training was demonstrated by DeDLOC (Diskin et al., 2021) and CocktailSGD (Wang et al., 2023) for training over slow networks. Recent work has focused on data-parallel pre-training over the internet: DiLoCo-style methods (Douillard et al., 2023; Sarfi et al., 2025; Jaghouar et al., 2024; Douillard et al., 2025) adapt federated-averaging (Stich, 2019) to drastically reduce synchronization frequency. DeMo (Peng et al., 2024) exploits fast orthonormal transforms, top-k sparsification, and error feedback to enable multi-datacenter training. ACCO (Nabli et al., 2024) overlaps the synchronization of delayed gradients with the computation of new gradients to increase GPU utilization. SparseLoCo (Sarfi et al., 2025) unifies error feedback, sparsification, and quantization to achieve internet-scale decentralized training (Lidin et al., 2026). However, all of these methods assume that each accelerator can hold a complete model replica. Pipeline-parallel training exhibits fundamentally different dynamics from DDP: SWARM (Ryabinin et al., 2023) identified the square-cube law, showing that computation scales cubically while communication scales quadratically, and prior work explored decentralized model parallelism via mixture-of-experts routing (Ryabinin & Gusev, 2020) and reversible architectures that decouple forward and backward passes across stages (Rivaud et al., 2025). Several works expanded on compressing activations (Chen et al., 2021; Bian et al., 2024) and the change in activations (Wang et al., 2022). Subspace Networks (Ramasinghe et al., 2025) exploited the empirical rank collapse of transformer residual streams to project boundary activations onto a fixed orthonormal basis. While effective, this imposes a single representational space across all transformer layers. In our work, we challenge this assumption to learn specialized, per-stage compression.

Efficient language models. A parallel line of work has investigated low-rank geometric structure for memory-efficient gradient compression and model compression, motivated by the empirical observation that deep networks converge to low-rank solutions (LeCun et al., 1989; Yu et al., 2017; Martin & Mahoney, 2021; Yang et al., 2023; Frankle & Carbin, 2019; Galanti et al., 2025). GaLore (Zhao et al., 2024) style methods (Chen et al., 2026; Lialin et al., 2024) projected stochastic gradients onto low-rank subspaces to reduce optimizer state memory, enabling full-parameter pre-training under tight memory budgets. Several works (Janson et al., 2026; Mo et al., 2025; Wei et al., 2024) further explored training low-rank factorized weights to reduce memory and communication requirements. LTE (Huh et al., 2024) used parallel LoRA-style adapters to pre-train models

from scratch with reduced memory and bandwidth across parallel accelerators. Pufferfish (Wang et al., 2021) modified the architecture to obtain trained factorized layers that reduce communication costs, but focused on data parallelism. Our work similarly modifies the transformer architecture, but with the complementary goal of enabling efficient inter-stage activation communication for pipeline parallelism.

3. Manifold and Dictionary Constrained Projection for Pipelined Parallelism

Figure 1 summarizes MDCP-PP. At each of the inter-stage boundaries, We subtract the token embeddings and project the boundary activations to a low-dimensional subspace via a learnable orthogonal projector, transmitted, and reconstructed at the receiving stage. We derive this design bottom-up — beginning from an empirical observation about the structure of boundary activations (§3.1), which motivates the construction (§3.2) and its optimization procedure (§3.3). We validate each component ablatively in §3.4 and sketch a composable vector-quantization extension in §3.5.

3.1. Boundary Activations are Intrinsically Low-Rank

Setup. Pipeline parallelism partitions the L layers of a decoder-only Transformer (Touvron et al., 2023b) into P contiguous stages $\mathcal{S}_1, \ldots, \mathcal{S}_P$, each residing on a dedicated device. Let E denote the embedding table and t_{ids} the input token indices. At each inter-stage boundary $p \in \{1, \ldots, P-1\}$, the forward pass communicates the boundary activation $X^{b_p} \in \mathbb{R}^{B\times T\times d}$, while the backward pass transmits its gradient; B, T, and d denote batch size, sequence length, and hidden dimension, respectively.

Motivating example. We profile a 150M-parameter LLaMA model (Touvron et al., 2023a) trained for 3,000 steps on DCLM (Li et al., 2024) with the Muon optimizer (Jordan et al., 2024) at P=8 pipeline stages. To isolate representational structure of the residual stream, we subtract the token embeddings and analyze the residual $X_{res} = X^{b_p} - E[t_{\text{ids}}]$, reshaped to $(B{\cdot}T) \times d$, via singular value decomposition (SVD). We quantify spectral concentration through the cumulative energy ratio:

$$\mathcal{E}(r) = \frac{\sum_{i=1}^{r} \sigma_i^2}{\sum_{i=1}^{d} \sigma_i^2}.$$

As shown in Figure 2, boundary activations exhibit pronounced low-rank structure across all pipeline stages: a rank of $r \approx 250$ suffices to retain $\geq$99% of the total activation energy, despite the ambient dimension d=1024. The inter-stage signal therefore concentrates near a submanifold of effective dimension roughly one quarter of the full representation space.

This finding extends (Ramasinghe et al., 2025), which observed rank collapse in projection matrices and responded by explicitly constraining weight rows to a shared low-rank subspace. Crucially, our setting requires *no* such constraint on weights: the low-rank structure emerges organically during training, without regularization or weight constraints. We therefore define a rank-r projector ($r < d$) as *information-preserving* for this signal, and argue that a learned projection can *discover*—rather than impose—the latent subspace the activations already inhabit, enabling principled compression of inter-stage communication.

3.2. Compression via Per-Stage Learned Orthogonal Projectors

We introduce, at each pipeline-stage boundary p, a learnable orthogonal projector A_p that lies on the Stiefel manifold

$$\mathrm{St}(d, r) = \{A \in \mathbb{R}^{d\times r} : A^\top A = I_r\}.$$

We equip each boundary with a per-stage anchor $E_p^{\text{small}}[t_{ids}]$ and a companion projector P_p that absorbs the high-rank, token-driven offset of the residual stream. Forward compression and reconstruction then proceed as

$$Z^{b_p} = \left(X^{b_p} - E_p^{\text{small}}[t_{ids}]P_p\right) A_p \ \in \ \mathbb{R}^{B\times T\times r}, \quad (1)$$

$$\hat{X}^{b_p} = Z^{b_p}\, A_p^\top + E_{p+1}^{\text{small}}[t_{ids}]P_{p+1} \ \in \ \mathbb{R}^{B\times T\times d}. \quad (2)$$

By enforcing the Stiefel constraint, we guarantee that $A_p^\top$ is the exact inverse of A_p. Projection and reconstruction are therefore isometric on the column space of A_p and the method achieves an exact compression ratio of r/d. In the backward pass we symmetrically route the gradient $\partial\mathcal{L}/\partial X^{b_p}$ through the same orthogonal projector A_p.

The rank-r projector A_p captures the dominant low-rank component of the boundary activation X^{b_p} while deliberately leaving the token-frequency offset unaddressed; this offset is inherently high-rank and would otherwise exhaust projector capacity. In SSN (Ramasinghe et al., 2025), the embedding table is decoupled into static high rank and learnable low rank components. Each stage added a static high rank offset as the learnable token embeddings forced to be in the same subspace as the weights (Ramasinghe et al., 2025). We instead add a learnable offset. To reduce the parameter pressure at each stage, we factorize the offset as

$$E_p^{\text{small}}[ids]\, P_p, \qquad E_p^{\text{small}} \in \mathbb{R}^{V\times r}, \qquad P_p \in \mathbb{R}^{r\times d}, \quad (3)$$

where E_p^{small} is a trainable embedding table and P_p is a fixed random orthonormal matrix. Only the integer token IDs cross the communication channel for this reconstruction. Consequently, the anchor adds negligible bandwidth overhead.

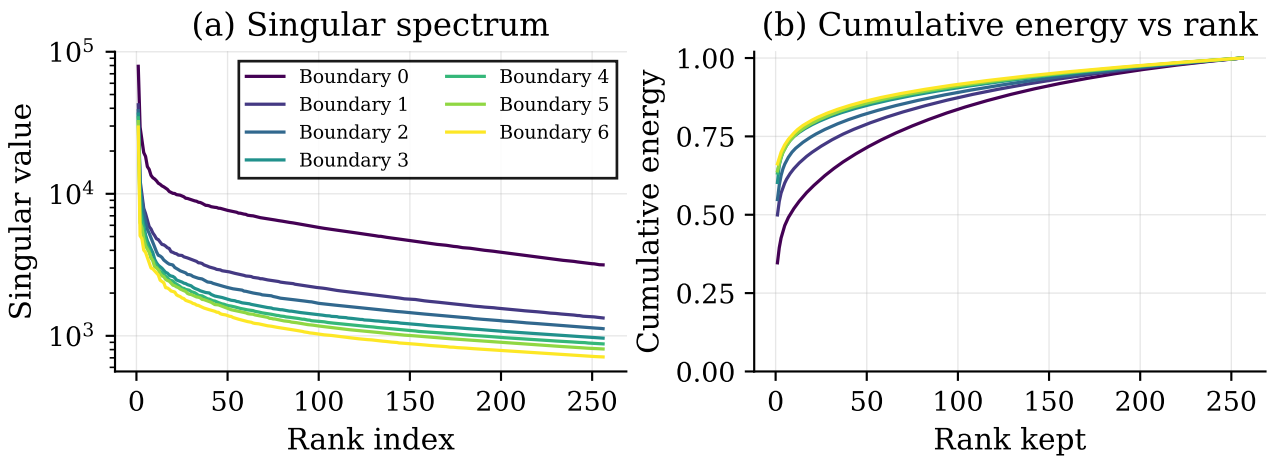

Figure 2. **Boundary activations exhibit intrinsic low-rank structure across all pipeline stages, with rank-250 truncation retaining $\geq$99% of activation energy.** (a) Singular value spectra of the centered boundary activations $X^{b_p} - E[t_{\text{ids}}]$ (reshaped to $(B \cdot T) \times d$) for all $P - 1 = 7$ inter-stage boundaries of a 150M LLaMA model ($d = 1024$, $P = 8$) trained with Muon (Jordan et al., 2024) on DCLM (Li et al., 2024). The x-axis indexes singular values in descending order; the y-axis is on a log scale. Each colored line corresponds to a distinct pipeline boundary. (b) Cumulative energy $\frac{\sum_{i=1}^{r}\sigma_i^2}{\sum_{i=1}^{d}\sigma_i^2}$ as a function of retained rank r. Color coding is shared across both panels.

3.3. Optimization via SPEL

We update each A_p jointly with the model weights via SPEL (Spectral Steepest Descent on the Stiefel Manifold) (Yang & Lai, 2026), using the task loss as the only signal. Given the Euclidean gradient $g_t = \partial\mathcal{L}/\partial A_p$ at step t and learning rate α, update consists of:

$$g_t^R = g_t - A_p\,\mathrm{sym}(A_p^\top g_t), \qquad \text{(tangent projection)} \tag{4}$$

$$m_t = \beta\, m_{t-1} + (1-\beta)\, g_t^R, \qquad \text{(heavy-ball momentum)} \tag{5}$$

$$d_t = \mathrm{PolarExpress}(m_t), \qquad \text{(spectral LMO direction)} \tag{6}$$

$$A_p \leftarrow \mathrm{PolarExpress}\big(A_p - \alpha\, d_t\big), \qquad \text{(retraction onto } \mathrm{St}(d,r)) \tag{7}$$

where $\mathrm{sym}(M) = \frac{1}{2}(M + M^\top)$. Polar Express (Amsel et al., 2026) calculates the spectral norm LMO direction which is a similar parameter update as Muon (Jordan et al., 2024) and retracts the matrix back to the Stiefel manifold. SPEL inherits the $\mathcal{O}(1/\sqrt{T})$ convergence rate of first-order methods on smooth manifolds (Yang & Lai, 2026). We send the updated A_p to the next stage after every optimizer step. This is a minimal communication cost when compared with activation and gradient communication.

Unlike SSN's Grassmann update that refreshes every $\sim$500 steps (Ramasinghe et al., 2025), SPEL updates A_p at every step, letting the subspace track the evolving activation geometry. In practice we choose a learning rate an order smaller than parameter updates ($\times 0.1$). Algorithm 1 summarizes the full per-boundary update.

Algorithm 1 MDCP-PP: per-boundary compression and projector update.

Require: Boundary activation X^{b_p}, projector $A_p \in \mathrm{St}(d,r)$, momentum buffer m_p, anchor E_p^{small}, companion projector P_p, token IDs t_{ids}, projector learning rate α, momentum β.

Project (sender): $Z^{b_p} \leftarrow \big(X^{b_p} - E_p^{\text{small}}[t_{ids}]\, P_p\big)\, A_p$
Transmit Z^{b_p} and t_{ids}
Reconstruct (receiver): $\hat{X}^{b_p} \leftarrow Z^{b_p}\, A_p^\top + E_{p+1}^{\text{small}}[t_{ids}]\, P_{p+1}$
Projector update (SPEL):
  $g \leftarrow \partial\mathcal{L}/\partial A_p$
  $g_R \leftarrow g - A_p\,\mathrm{sym}(A_p^\top g)$ {tangent projection}
  $m_p \leftarrow \beta\, m_p + (1-\beta)\, g_R$ {momentum}
  $d \leftarrow \mathrm{PolarExpress}(m_p)$ {LMO direction}
  $A_p \leftarrow \mathrm{PolarExpress}(A_p - \alpha\, d)$ {retraction}

3.4. Empirical Validation

We validate three properties of our method (§3.2) and show learning the projection is naturally more favorable compression than having a fixed projector. We show that per-boundary projectors discover geometrically distinct subspaces for each different stage. The learned A_p captures substantially more activation energy than a fixed orthonormal basis of equal rank, and the learned projection faithfully preserves pairwise token similarity after compression.

Each stage learns a distinct subspace. Figure 3 (a) re-

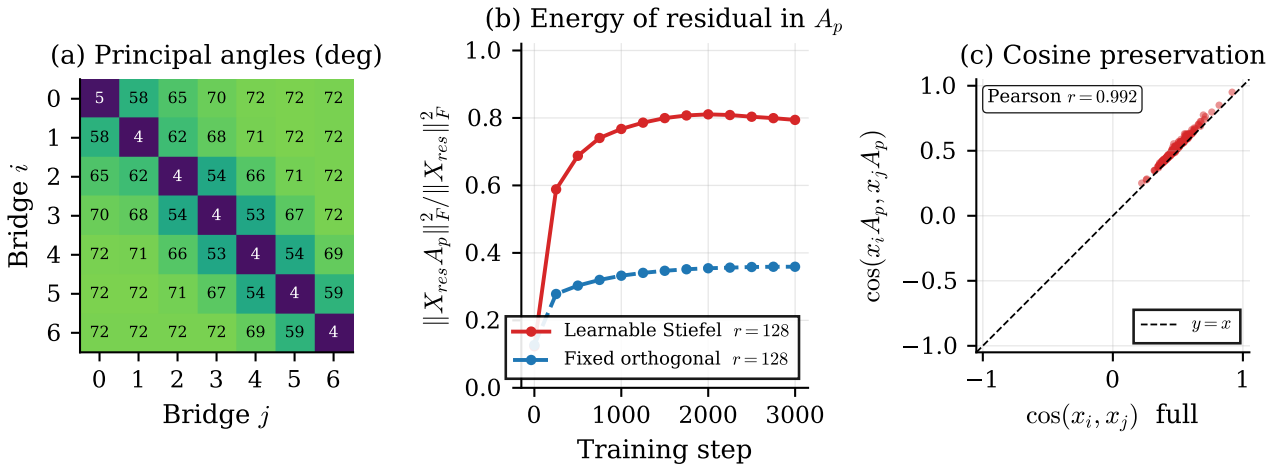

Figure 3. **Empirical validation of learned orthogonal projectors for activation compression in pipeline-parallel training.** (a) Pairwise mean principal angles (degrees) between learned Stiefel manifold projectors across pipeline stages i and j, computed via $\arccos(\sigma_k(A_i^\top A_j))$. Large off-diagonal angles (up to 72°) confirm that projectors across non-adjacent stages converge to geometrically distinct subspaces, while near-zero diagonal values verify self-consistency. (b) Frobenius-norm energy of residual activations preserved under projection, $\|X_{\text{res}}A_p\|_F^2/\|X_{\text{res}}\|_F^2$, as a function of training step for learned Stiefel ($r = 128$, red) versus fixed orthogonal ($r = 128$, blue) projectors. Learned projectors retain approximately 2.2× more residual energy (~ 0.80 vs. ~ 0.36), demonstrating superior task-adaptive compression. (c) Cosine similarity preservation between token pairs: $\cos(x_iA_p,\ x_jA_p)$ versus $\cos(x_i, x_j)$ evaluated on the full-dimensional representations. A Pearson correlation of $r = 0.992$ confirms that the learned projection A_p faithfully preserves the pairwise relational geometry of the activation space, closely tracking the identity line ($y = x$).

ports pairwise mean principal angles between the seven learned projectors of all boundaries, computed as $\bar{\theta} = \frac{1}{r}\sum_{k=1}^{r}\arccos\big(\sigma_k(A_i^\top A_j)\big)$. Off-diagonal angles range from 53° to 72°, with adjacent stages exhibiting the greatest overlap and distant stages approaching near-orthogonality. This change in subspace alignment is consistent with the residual stream transitioning from lexical to task-specific representations across depth (Zeiler & Fergus, 2014).

The learned projector captures substantially more energy. Figure 3 (b) compares the residual energy retained under projection, $\|X_{\text{res}}A_p\|_F^2/\|X_{\text{res}}\|_F^2$, between a fixed random orthonormal basis and the jointly trained A_p at matching rank r=128. Note that $\|A\|_F^2 = \sum_i \sigma_i^2$. The fixed basis plateaus near 36% of captured energy while the learned projector reaches ∼80% within 1,500 steps and continues to improve—a factor-of-two gain in effective rank utilization. Inspection of weight stable rank confirms that the preceding attention out projection and MLP projection layers adapt to this space defined by (A_p) (Appendix E). We see that the compression actively induces the low rank structure.

The learned projection preserves pairwise token geometry. Figure 3 (c) plots the cosine similarity between projected token pairs, $\cos(x_iA_p,\ x_jA_p)$, against their full-dimensional counterpart $\cos(x_i, x_j)$. The Pearson correlation of $r = 0.992$ with the identity line demonstrates that A_p behaves as a near-isometry over the empirical token distribution and the relative angular structure in the activation space is preserved under compression. This property is not enforced explicitly and it emerges from Stiefel optimization alone and provides geometric evidence that the transmitted representation retains the relational information downstream stages require for computation.

3.5. Composing with Vector Quantization

To minimize communication overhead during pipeline-parallel training, we employ Multi-Codebook Vector Quantization (MCVQ) for compressing both forward activations and backward gradients. The method operates on a low-rank projected representation, decomposed into G groups, and applies R rounds of residual quantization to achieve high compression ratios while preserving representational fidelity.

Quantization Scheme. Formally, the projected representation is compressed via residual vector quantization using a per-stage codebook. $Z^{b_p} \in \mathbb{R}^{B\times T\times r}$ is compressed using a per-stage codebook $\mathcal{C}_p \in \mathbb{R}^{r\times K}$. Each stage quantizes the residual error from the previous round, enabling progressive refinement of the compressed representation across R rounds.

Codebook Synchronization. Sender and receiver nodes synchronize $\mathcal{C}_p$ through a streaming dictionary update protocol: the codebook is partitioned into random subsets, with a $\frac{1}{K}$ fraction of codes transmitted per micro-batch. This design is motivated by the empirical observation of (Zhang

et al., 2024) that VQ codebooks evolve slowly over the course of training, ensuring that the staleness introduced by incremental updates remains negligible with respect to overall bandwidth and convergence.

4. Experiments

4.1. Experimental Setup

Architecture and scales. We evaluate our method on decoder-only transformers based on the LLaMA (Touvron et al., 2023a) architecture across three parameter scales: 150M, 500M, and 1B. All models employ a context length of 2048 tokens and are trained with the LLaMA tokenizer (vocabulary size 32,000) in bf16 precision. All models are pre-trained on the DCLM corpus (Li et al., 2024), with 5M tokens held out uniformly at random as a validation set. Following the Chinchilla compute-optimal scaling regime (Hoffmann et al., 2022), each model is trained on a token budget of 20 tokens per parameter. Implementation details are provided in Appendix F. All the methods are compared at equal token budgets.

Optimization. We adopt a hybrid optimizer configuration in which the 2D hidden weight matrices are updated using Muon (Jordan et al., 2024), while embeddings, biases, and the output projection are updated with AdamW (Loshchilov & Hutter, 2019). The Muon learning rate is set to $\eta_\mu = 0.02$ for the 150M, 500M sizes and $\eta_\mu = 0.01$ for the 1B case, and the AdamW learning rate is coupled to it via a multiplier of 0.5, yielding $\eta_{\text{adam}} = 0.5 \cdot \eta_\mu$. Training uses a global batch size of 512. We choose the multiplier of 0.1 for learning rate we use to update the projector using § 3.3. Our primary experiments are conducted under pipeline-parallel configurations with $P \in \{4, 8\}$ stages.

Baselines. We compare against methods while holding all shared hyperparameters fixed to ensure a fair comparison. **Uncompressed** transmits activations and gradients across pipeline boundaries at the full model width and therefore serves as an upper bound on attainable quality. **SSN** (Ramasinghe et al., 2025) projects inter-stage tensors onto a learned low-rank subspace (§ B). **SSN (AdamW version)** uses SSN with AdamW optimizer for all the parameters. **MDCP-PP** applies our proposed compression scheme to inter-stage activations and gradients. Finally, **MDCP-PP+ VQ** augments our approach with vector quantization of the projected representation, effectively doubling the compression ratio.

Evaluation. We report cross-entropy validation loss on the held-out DCLM split as our primary metric, together with the relative degradation $\Delta\%$ with respect to the Uncompressed baseline at the same scale. We additionally evaluate zero-shot downstream accuracy on HellaSwag (Zellers et al., 2019), PIQA (Bisk et al., 2020), ARC-Easy, and ARC-Challenge (Clark et al., 2018) using `lm-evaluation-harness` (Gao et al., 2024). We also report average accuracy across all the tasks.

4.2. Main Results

MDCP-PP closes the gap to the uncompressed baseline across all scales. Table 1 shows that, at $4\times$ compression on the 150M model, MDCP-PP achieves validation losses of 3.156 (P=4) and 3.165 (P=8). This is just 0.84% and 1.11% above the uncompressed reference of 3.13. At this scale, we can see that SSN degrades much higher in performance in both pipeline cases. Scaling to 500M parameters under $6\times$ compression, MDCP-PP tracks the baseline within 1.90% (P=4) and 0.08% (P=8); at 1B under $8\times$ compression, it holds losses of 2.72 and 2.73, degrading by only 1.38% and 2.02% respectively. At all scales the SSN performs much worse and loses a maximum of nearly 14% performance compared to uncompressed baseline.

We attribute the stability of MDCP-PP across model scales to the per-stage Stiefel-constrained projectors, which faithfully capture the intrinsic low-rank geometry of the activations identified in §3.1. Composing MDCP-PP with residual vector quantization (§3.5) doubles the compression ratio while incurring only marginal additional loss. At 1B parameters, MDCP-PP+VQ achieves $16\times$ inter-stage compression at 3.01% (P=4) and 2.30% (P=8) degradation. We trace this behavior to the empirical near-isometry of the learned projector (Figure 3c). By preserving pairwise similarity on the low-rank manifold, the codebook in $\mathbb{R}^r$ inherits a well-conditioned distribution that aligns itself naturally for vector quantization.

MDCP-PP preserves downstream performance on various tasks. Validation-loss improvements translate directly to zero-shot performance as shown in Table 2. Across the tested downstream tasks, MDCP-PP closely tracks the uncompressed baseline at every scale. At 500M parameters, the average accuracy gap is 0.2 points at P=4 (41.8 vs. 42.0) and 0.4 points at P=8 (41.6 vs. 42.0); at 1B, the corresponding gaps are 0.8 and 1.5 average points. SSN baselines, by contrast, suffer accuracy drops as large as 8.8 points at 1B. This shows that the learning capacity is restricted due to the global-subspace weight constraint and shows degraded performance at token matched comparisons. The MDCP-PP+VQ variant incurs a more pronounced performance loss in downstream tasks.

5. Conclusion

We introduced **MDCP-PP**, a communication-efficient pipeline-parallel training method that learns stage-specific orthogonal compression subspaces directly on the Stiefel manifold using manifold-constrained optimization. By

Scale	Method	Bytes/token	Comp.	P=4 Loss	P=4 Δ %	P=8 Loss	P=8 Δ %
150M	*Uncompressed*	2048	—	3.13	—	3.13	—
150M	SSN (Ramasinghe et al., 2025) (AdamW (Kingma & Ba, 2015) version)	512	4×	3.49	11.49%	3.52	12.46%
150M	SSN (Ramasinghe et al., 2025)	512	4×	3.39	8.37%	3.40	8.63%
150M	MDCP-PP	512	4×	3.156	**0.84%**	3.165	**1.11%**
150M	MDCP-PP+ VQ	256	8×	3.165	1.11%	3.170	1.28%
500M	*Uncompressed*	3072	—	2.84	—	2.84	—
500M	SSN (Ramasinghe et al., 2025) (AdamW (Kingma & Ba, 2015) version)	512	6×	3.25	14.34%	3.29	15.90%
500M	SSN (Ramasinghe et al., 2025)	512	6×	3.09	8.92%	3.12	9.90%
500M	MDCP-PP	512	6×	2.79	**-1.90%**	2.84	**0.00%**
500M	MDCP-PP+ VQ	256	12×	2.92	2.75%	2.88	1.49%
1B	*Uncompressed*	4096	—	2.68	—	2.68	—
1B	SSN (Ramasinghe et al., 2025) (AdamW (Kingma & Ba, 2015) version)	512	8×	3.38	26.34%	3.39	26.42%
1B	SSN (Ramasinghe et al., 2025)	512	8×	3.05	13.93%	3.08	15.05%
1B	MDCP-PP	512	8×	2.72	**1.38%**	2.73	**2.02%**
1B	MDCP-PP+ VQ	256	16×	2.76	3.01%	2.74	2.30%

Table 1. **MDCP-PP closes the performance gap to uncompressed training across all model scales while delivering 4–16× communication compression.** Cross entropy validation loss and compression results for 150M, 500M, and 1B parameter models under pipeline-parallel settings with P=4 and P=8 show that both MDCP-PP and its vector-quantized variant consistently match or approach the uncompressed baseline. In contrast, they substantially outperform the SSN baselines in accuracy. Best compressed values are marked with **bold** text.

Size	Config	P=4					P=8				
		HellaSwag	PIQA	ARC-E	ARC-C	Avg	HellaSwag	PIQA	ARC-E	ARC-C	Avg
150M	*Uncompressed*	28.1	60.1	38.2	22.8	37.3	28.1	60.1	38.2	22.8	37.3
	SSN (Ramasinghe et al., 2025) (AdamW (Kingma & Ba, 2015) version)	26.5	56.1	32.5	20.6	33.9	26.8	56.0	32.1	20.6	33.9
	SSN (Ramasinghe et al., 2025)	27.6	56.7	34.4	20.9	34.9	27.0	56.3	33.8	21.7	34.7
	MDCP-PP	28.0	59.4	37.0	22.6	**36.7**	27.8	59.3	37.8	21.5	**36.6**
	MDCP-PP + VQ	28.4	52.8	31.7	22.7	33.9	26.4	50.5	29.1	25.9	33.0
500M	*Uncompressed*	35.7	64.4	43.2	24.8	42.0	35.7	64.4	43.2	24.8	42.0
	SSN (Ramasinghe et al., 2025) (AdamW (Kingma & Ba, 2015) version)	27.0	59.5	35.6	21.2	35.8	27.4	57.9	34.6	22.1	35.5
	SSN (Ramasinghe et al., 2025)	29.1	59.8	38.5	22.3	37.4	28.5	59.1	37.2	22.2	36.7
	MDCP-PP	35.1	64.0	43.6	24.3	**41.8**	34.5	64.7	42.4	24.7	**41.6**
	MDCP-PP + VQ	25.9	50.8	26.6	26.7	32.5	26.9	51.2	27.2	24.3	32.4
1B	*Uncompressed*	38.8	66.3	46.8	24.5	44.1	38.8	66.3	46.8	24.5	44.1
	SSN (Ramasinghe et al., 2025) (AdamW (Kingma & Ba, 2015) version)	27.5	57.9	35.1	20.8	35.3	27.4	57.6	35.6	20.7	35.3
	SSN (Ramasinghe et al., 2025)	29.1	61.4	37.1	22.7	37.6	28.7	60.6	37.2	21.8	37.1
	MDCP-PP	37.1	65.7	45.7	24.5	**43.3**	36.2	64.8	45.2	24.1	**42.6**
	MDCP-PP + VQ	25.9	49.6	26.6	27.6	32.4	25.8	49.8	27.1	28.0	32.7

Table 2. **Zero-shot downstream benchmark accuracy under pipeline-parallel communication compression.** Accuracy on HellaSwag, PIQA, ARC-Easy (ARC-E), and ARC-Challenge (ARC-C) for 150M, 500M, and 1B parameter models trained with P=4 and P=8 pipeline stages. MDCP-PP consistently approaches the uncompressed baseline across all benchmarks and scales— averaging within ∼1 point at 500M and 1B—while SSN baselines suffer larger accuracy drops (up to ∼9 average points at 1B scale). The vector-quantized variant (MDCP-PP + VQ) achieves higher compression ratios but at a notable accuracy cost. Averages (Avg) are reported across all four tasks. The best compressed values are marked with bold text.

combining adaptive low-rank projectors with lightweight factorized anchor embeddings and optional residual vector quantization, MDCP-PP enables aggressive inter-stage activation compression while preserving training quality. Across LLaMA-style models ranging from 150M to 1B parameters and pipeline configurations with 4 and 8 stages, MDCP-PP achieves $4\times$–$8\times$ communication compression within approximately 1–2% of the uncompressed validation loss. It extends to $16\times$ compression with only modest additional degradation when combined with vector quantization. The method consistently outperforms recent work in subspace networks (Ramasinghe et al., 2025) while maintaining strong downstream zero-shot accuracy. Our results suggest that inter-stage transformer activations naturally admit adaptive low-dimensional structure that can be learned jointly with the task objective, rather than imposed through a fixed global subspace. More broadly, this work highlights manifold-constrained representation learning as a practical mechanism for reducing communication costs in distributed foundation-model training.

Limitations. Our experiments are limited to models up to 1B parameters, and further evaluation at larger scales and under real heterogeneous network conditions remains necessary. In addition, although MDCP-PP substantially reduces the degradation associated with activation compression, performance still declines under extreme compression ratios, particularly when vector quantization is applied.

References

Alistarh, D., Grubic, D., Li, J., Tomioka, R., and Vojnovic, M. Qsgd: Communication-efficient sgd via gradient quantization and encoding. *Advances in neural information processing systems*, 30, 2017.

Amsel, N., Persson, D., Musco, C., and Gower, R. M. The polar express: Optimal matrix sign methods and their application to the muon algorithm. *International Conference on Learning Representations (ICLR)*, 2026.

Bernstein, J., Wang, Y.-X., Azizzadenesheli, K., and Anandkumar, A. signsgd: Compressed optimisation for non-convex problems. In *International conference on machine learning*, pp. 560–569. PMLR, 2018.

Bian, S., Li, D., Wang, H., Xing, E. P., and Venkataraman, S. Does compressing activations help model parallel training? *Proceedings of Machine Learning and Systems*, 6:239–252, 2024.

Bisk, Y., Zellers, R., Gao, J., Choi, Y., et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 7432–7439, 2020.

Chen, J., Zheng, L., Yao, Z., Wang, D., Stoica, I., Mahoney, M., and Gonzalez, J. Actnn: Reducing training memory footprint via 2-bit activation compressed training. In *International Conference on Machine Learning*, pp. 1803–1813. PMLR, 2021.

Chen, X., Feng, K., Li, C., Lai, X., Yue, X., Yuan, Y., and Wang, G. Fira: Can we achieve full-rank training of LLMs under low-rank constraint? In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=7aSBAw7tJf.

Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., and Tafjord, O. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Dettmers, T., Lewis, M., Shleifer, S., and Zettlemoyer, L. 8-bit optimizers via block-wise quantization. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=shpkpVXzo3h.

Diskin, M., Bukhtiyarov, A., Ryabinin, M., Saulnier, L., Sinitsin, A., Popov, D., Pyrkin, D. V., Kashirin, M., Borzunov, A., Villanova del Moral, A., et al. Distributed deep learning in open collaborations. *Advances in Neural Information Processing Systems*, 34:7879–7897, 2021.

Douillard, A., Feng, Q., Rusu, A. A., Chhaparia, R., Donchev, Y., Kuncoro, A., Ranzato, M., Szlam, A., and Shen, J. Diloco: Distributed low-communication training of language models. *CoRR*, abs/2311.08105, 2023. URL https://doi.org/10.48550/arXiv.2311.08105.

Douillard, A., Donchev, Y., Rush, J. K., Kale, S., Charles, Z., Teston, G., Garrett, Z., Shen, J., McIlroy, R., Lacey, D., Rame, A., Szlam, A., Ranzato, M., and Barham, P. R. Streaming diloco with overlapping communication. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=yYk3zK0X6Q.

Frankle, J. and Carbin, M. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJl-b3RcF7.

Galanti, T., Siegel, Z. S., Gupte, A., and Poggio, T. A. Sgd with weight decay secretly minimizes the ranks of your neural networks. In *The Second Conference on Parsimony and Learning (Proceedings Track)*, 2025.

Gao, L., Tow, J., Abbasi, B., Biderman, S., Black, S., DiPofi, A., Foster, C., Golding, L., Hsu, J., Le Noac'h, A., Li, H., McDonell, K., Muennighoff, N., Ociepa, C., Phang, J., Reynolds, L., Schoelkopf, H., Skowron, A., Sutawika, L., Tang, E., Thite, A., Wang, B., Wang, K., and Zou, A. The language model evaluation harness, 07 2024. URL https://zenodo.org/records/12608602.

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., Casas, D. d. L., Hendricks, L. A., Welbl, J., Clark, A., et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Huang, Y., Cheng, Y., Bapna, A., Firat, O., Chen, D., Chen, M., Lee, H., Ngiam, J., Le, Q. V., Wu, Y., et al. Gpipe: Efficient training of giant neural networks using pipeline parallelism. *Advances in neural information processing systems*, 32, 2019.

Huh, M., Cheung, B., Bernstein, J., Isola, P., and Agrawal, P. Training neural networks from scratch with parallel low-rank adapters. *arXiv preprint arXiv:2402.16828*, 2024.

Jaghouar, S., Ong, J. M., and Hagemann, J. Opendiloco: An open-source framework for globally distributed low-communication training, July 2024. URL https://arxiv.org/abs/2407.07852.

Janson, P., Oyallon, E., and Belilovsky, E. Stabilizing native low-rank LLM pretraining. In *Forty-third International*

Conference on Machine Learning, 2026. URL https://openreview.net/forum?id=kb5wjkqu8q.

Jordan, K., Jin, Y., Boza, V., You, J., Cesista, F., Newhouse, L., and Bernstein, J. Muon: An optimizer for hidden layers in neural networks, 2024. URL https://web.archive.org/web/20250122060345/https://kellerjordan.github.io/posts/muon/.

Karimireddy, S. P., Rebjock, Q., Stich, S., and Jaggi, M. Error feedback fixes signsgd and other gradient compression schemes. In *International conference on machine learning*, pp. 3252–3261. PMLR, 2019.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015. URL https://arxiv.org/abs/1412.6980.

Koloskova, A., Stich, S., and Jaggi, M. Decentralized stochastic optimization and gossip algorithms with compressed communication. In *International conference on machine learning*, pp. 3478–3487. PMLR, 2019.

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

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=H1eA7AEtvS.

LeCun, Y., Denker, J., and Solla, S. Optimal brain damage. *Advances in neural information processing systems*, 2, 1989.

Li, J., Fang, A., Smyrnis, G., Ivgi, M., Jordan, M., Gadre, S. Y., Bansal, H., Guha, E. K., Keh, S. S., Arora, K., Garg, S., Xin, R., Muennighoff, N., Heckel, R., Mercat, J., Chen, M. F., Gururangan, S., Wortsman, M., Albalak, A., Bitton, Y., Nezhurina, M., Abbas, A., Hsieh, C., Ghosh, D., Gardner, J., Kilian, M., Zhang, H., Shao, R., Pratt, S. M., Sanyal, S., Ilharco, G., Daras, G., Marathe, K., Gokaslan, A., Zhang, J., Chandu, K. R., Nguyen, T., Vasiljevic, I., Kakade, S. M., Song, S., Sanghavi, S., Faghri, F., Oh, S., Zettlemoyer, L., Lo, K., El-Nouby, A., Pouransari, H., Toshev, A., Wang, S., Groeneveld, D., Soldaini, L., Koh, P. W., Jitsev, J., Kollar, T., Dimakis, A., Carmon, Y., Dave, A., Schmidt, L., and Shankar, V. Datacomp-lm: In search of the next generation of training sets for language models. In Globersons, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J. M., and Zhang, C. (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/19e4ea30dded58259665db375885e412-Abstract-Datasets_and_Benchmarks_Track.html.

Lialin, V., Muckatira, S., Shivagunde, N., and Rumshisky, A. ReloRA: High-rank training through low-rank updates. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=DLJznSp6X3.

Lian, X., Zhang, C., Zhang, H., Hsieh, C.-J., Zhang, W., and Liu, J. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. *Advances in neural information processing systems*, 30, 2017.

Lidin, J., Sarfi, A., Miahi, E., Anthony, Q., Chauhan, S., Pappas, E., Thérien, B., Belilovsky, E., and Dare, S. Covenant-72b: Pre-training a 72b llm with trustless peers over-the-internet. *arXiv preprint arXiv:2603.08163*, 2026.

Lin, Y., Han, S., Mao, H., Wang, Y., and Dally, B. Deep gradient compression: Reducing the communication bandwidth for distributed training. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=SkhQHMW0W.

Liu, J., Su, J., Yao, X., Jiang, Z., Lai, G., Du, Y., Qin, Y., Xu, W., Lu, E., Yan, J., Chen, Y., Zheng, H., Liu, Y., Liu, S., Yin, B., He, W., Zhu, H., Wang, Y., Wang, J., Dong, M., Zhang, Z., Kang, Y., Zhang, H., Xu, X., Zhang, Y., Wu, Y., Zhou, X., and Yang, Z. Muon is scalable for llm training, February 2025. URL https://arxiv.org/abs/2502.16982.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

Martin, C. H. and Mahoney, M. W. Implicit self-regularization in deep neural networks: Evidence from random matrix theory and implications for learning. *Journal of Machine Learning Research*, 22(165):1–73, 2021.

Mo, Z., Huang, L.-K., and Pan, S. J. Parameter and memory efficient pretraining via low-rank riemannian optimization. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=i0zzO7Hslk.

Nabli, A., Fournier, L., Erbacher, P., Serrano, L., Belilovsky, E., and Oyallon, E. Acco: Accumulate while you communicate for communication-overlapped sharded llm training. *arXiv preprint arXiv:2406.02613*, 2024.

Nabli, A., Fournier, L., Erbacher, P., Serrano, L., Belilovsky, E., and Oyallon, E. Acco: Accumulate while you communicate for communication-overlapped sharded llm training. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Peng, B., Quesnelle, J., and Kingma, D. P. Decoupled momentum optimization. *arXiv preprint arXiv:2411.19870*, 2024.

Rajbhandari, S., Rasley, J., Ruwase, O., and He, Y. Zero: Memory optimizations toward training trillion parameter models. In *SC20: international conference for high performance computing, networking, storage and analysis*, pp. 1–16. IEEE, 2020.

Ramasinghe, S., Ajanthan, T., Avraham, G., Zuo, Y., and Long, A. Subspace networks: Scaling decentralized training with communication-efficient model parallelism. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=kke9TwtKi0.

Rivaud, S., Fournier, L., Pumir, T., Belilovsky, E., Eickenberg, M., and Oyallon, E. PETRA: Parallel end-to-end training with reversible architectures. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=0fhzSFsGUT.

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

Ryabinin, M., Dettmers, T., Diskin, M., and Borzunov, A. Swarm parallelism: Training large models can be surprisingly communication-efficient. In *International Conference on Machine Learning*, pp. 29416–29440. PMLR, 2023.

Sarfi, A., Thérien, B., Lidin, J., and Belilovsky, E. Communication efficient llm pre-training with sparseloco. *arXiv preprint arXiv:2508.15706*, 2025.

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

Su, J., Ahmed, M., Lu, Y., Pan, S., Bo, W., and Liu, Y. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Tang, H., Gan, S., Awan, A. A., Rajbhandari, S., Li, C., Lian, X., Liu, J., Zhang, C., and He, Y. 1-bit adam: Communication efficient large-scale training with adam's convergence speed. In *International Conference on Machine Learning*, pp. 10118–10129. PMLR, 2021.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023a.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023b.

Vogels, T., Karimireddy, S. P., and Jaggi, M. Powersgd: Practical low-rank gradient compression for distributed optimization. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 14236–14245, 2019a. URL https://proceedings.neurips.cc/paper/2019/hash/d9fbed9da256e344c1fa46bb46c34c5f-Abstract.html.

Vogels, T., Karimireddy, S. P., and Jaggi, M. Powersgd: Practical low-rank gradient compression for distributed optimization. *Advances in Neural Information Processing Systems*, 32, 2019b.

Wang, H., Agarwal, S., and Papailiopoulos, D. Pufferfish: Communication-efficient models at no extra cost. *Proceedings of Machine Learning and Systems*, 3:365–386, 2021.

Wang, J., Kolar, M., Srebro, N., and Zhang, T. Efficient distributed learning with sparsity. In *International conference on machine learning*, pp. 3636–3645. PMLR, 2017.

Wang, J., Yuan, B., Rimanic, L., He, Y., Dao, T., Chen, B., Ré, C., and Zhang, C. Fine-tuning language models over slow networks using activation quantization with guarantees. *Advances in Neural Information Processing Systems*, 35:19215–19230, 2022.

Wang, J., Lu, Y., Yuan, B., Chen, B., Liang, P., De Sa, C., Re, C., and Zhang, C. Cocktailsgd: Fine-tuning foundation

models over 500mbps networks. In *International Conference on Machine Learning*, pp. 36058–36076. PMLR, 2023.

Wangni, J., Wang, J., Liu, J., and Zhang, T. Gradient sparsification for communication-efficient distributed optimization. *Advances in Neural Information Processing Systems*, 31, 2018.

Wei, X., Moalla, S., Pascanu, R., and Gulcehre, C. Building on efficient foundations: Effective training of llms with structured feedforward layers. In Globerson, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J., and Zhang, C. (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 4689–4717. Curran Associates, Inc., 2024. doi: 10.52202/079017-0153. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/0877af85978e9e630b77f6221db47876-Paper-C pdf.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

Wu, J., Huang, W., Huang, J., and Zhang, T. Error compensated quantized sgd and its applications to large-scale distributed optimization. In *International conference on machine learning*, pp. 5325–5333. PMLR, 2018.

Yang, G., Simon, J. B., and Bernstein, J. A spectral condition for feature learning. *arXiv preprint arXiv:2310.17813*, 2023.

Yang, K. and Lai, L. Manifold constrained steepest descent. *arXiv preprint arXiv:2601.21487*, 2026.

Yu, X., Liu, T., Wang, X., and Tao, D. On compressing deep models by low rank and sparse decomposition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7370–7379, 2017.

Zeiler, M. D. and Fergus, R. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pp. 818–833. Springer, 2014.

Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., and Choi, Y. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 4791–4800, 2019.

Zhang, B., Wang, H., Luo, C., Li, X., Liang, G., Ye, Y., Qi, X., and He, Y. Codebook transfer with part-of-speech for vector-quantized image modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7757–7766, 2024.

Zhao, J., Zhang, Z., Chen, B., Wang, Z., Anandkumar, A., and Tian, Y. Galore: Memory-efficient llm training by gradient low-rank projection, 2024. URL https://arxiv.org/abs/2403.03507.

A. Pareto Frontier

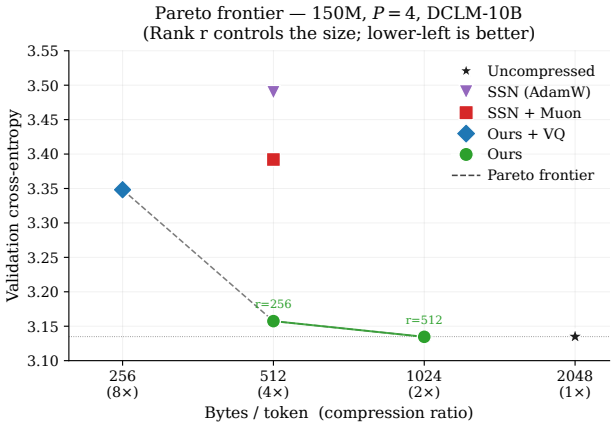

Figure 4. Validation cross-entropy versus communication cost (bytes per token) for pipeline-parallel training of a 150M-parameter model with $P = 4$ stages on DCLM-10B. Lower-left is better. We compare our learned projection approach ("Ours") and its vector-quantized variant ("Ours + VQ") against uncompressed training and SSN-based baselines. Our method traces the Pareto frontier across compression regimes, achieving substantially lower validation cross-entropy at comparable or smaller communication budgets. In particular, the VQ-enhanced variant reaches the strongest compression point ($8\times$) while remaining on the Pareto frontier, whereas the learned projection alone achieves near-uncompressed performance at $2\times$–$4\times$ compression.

Figure 4 shows the trade-off between communication cost and validation performance for pipeline-parallel training with four stages. Each point corresponds to a different activation communication size, reported in bytes per token relative to the uncompressed baseline. The proposed learned projection consistently outperforms SSN-based compression methods at comparable bandwidth budgets, indicating that adaptive learned projections preserve substantially more task-relevant information under aggressive communication constraints. Moreover, combining our approach with vector quantization extends the achievable compression regime to $8\times$ while remaining on the Pareto frontier. These results demonstrate that the proposed method provides a favorable accuracy–communication trade-off across a wide range of operating points.

B. Background: Subspace Networks

SSN (Ramasinghe et al., 2025) exploits a structural property of trained Transformers: the output projection weights $W_2^\ell \in \mathbb{R}^{d_{\text{ff}} \times d}$ and attention projection weights $W_1^\ell \in \mathbb{R}^{d \times d}$

undergo *rank collapse* during training, with their effective row space converging to a low-dimensional subspace $\mathcal{S} \subset \mathbb{R}^d$ of dimension $k \ll d$. This collapse arises naturally from AdamW's adaptive scaling, which attenuates gradient components aligned with negligible singular directions.

SSN operationalizes this observation via five steps. **(1)** A fixed orthonormal basis $U_k \in \mathbb{R}^{d\times k}$ is constructed by QR decomposition of a random matrix. **(2)** The rows of each W_2^ℓ and W_1^ℓ are constrained to $\mathrm{span}(U_k)$ by replacing the per-parameter adaptive scaling in AdamW with a row-wise constant variant. **(3)** Deterministic components—positional embeddings PE and fixed token embeddings T_{fixed}—are subtracted from X^{b_p} before compression. **(4)** The stage transmits the compressed activation

$$Z^{b_p} = \left(X^{b_p} - \mathrm{PE} - T_{\text{fixed}}\right) U_k \ \in \ \mathbb{R}^{B\times T\times k}, \quad (8)$$

achieving a compression ratio of k/d. **(5)** The receiving stage reconstructs $\hat{X}^{b_p} = Z^{b_p} U_k^\top + \mathrm{PE} + T_{\text{fixed}}$.

Because weight confinement ensures $X^{b_p} - \mathrm{PE} - T_{\text{fixed}} \in \mathrm{span}(U_k)$ exactly, SSN's compression is *architecturally lossless* on its modified model, though it incurs loss relative to an unconstrained baseline. The basis U_k is refined via Grassmann manifold updates every $\sim$500 steps.

Limitations. SSN's losslessness is contingent on strict weight confinement. The subspace is fixed at initialization and only coarsely adapted; the model is forced to conform to a predetermined geometry rather than discovering a task-optimal one. In §3.1, we showed that pipeline boundary activations exhibit low-rank structure *intrinsically*—even when trained with a modern optimizer (Jordan et al., 2024) without any architectural constraint—motivating a jointly learned subspace that adapts to the activation geometry rather than imposing it.

Table 3. Comparison of architectural features between Subspace Networks (SSN) and MDCP-PP. Our method enables high-ratio compression without constraining model weights to low-rank subspaces, while maintaining compatibility with diverse optimizers through Stiefel manifold learning.

Feature	SSN	MDCP-PP	+VQ
Unconstrained Full-Rank Weights	×	✓	✓
Learnable Projection Basis	Slow Grassmann	✓	✓
Decoupled Low-Rank Embeddings	✓	×	×
Learnable Per-Stage Anchor	×	✓	✓
Discrete Vector Quantization	×	×	✓

C. Evaluating SSN without Subspace Projection

We ran probe evaluations on 16 sequences of 2048 tokens drawn from the DCLM validation set (Li et al., 2024). In this ablation, we replaced the subspace projection matrix U_k with an identity matrix at inference time and replaced the decoupled low rank embedding table with its projections $TE \mathbin{@} U_k^\top$.

As a reference, the uniform-distribution baseline (i.e., random guessing over the vocabulary) yields a loss of ≈ 10.38. Any value above this threshold indicates performance worse than chance.

The learned subspace projection matrices U_k do more than compress information—they form an integral part of the model's learned representations. When we replace them with the identity matrix at inference time (while keeping all weights trained with active projection), the model suffers severe degradation across every tested configuration. Losses increase by between $+3.44$ and $+8.25$, and several $P = 8$ runs exceed the uniform baseline entirely.

This effect holds consistently across model scales (150M–1B), optimizers (Muon (Jordan et al., 2024) and AdamW (Kingma & Ba, 2015)), and degrees of parallelism. These results show that the network internalizes representations tightly coupled to the projected subspaces. Removing the projection does not simply degrade the model but breaks it.

Table 4. Loss with trained U_k versus identity ablation ($U_k = I$). Values above ≈ 10.38 (uniform random baseline) indicate worse-than-chance predictions.

Size	pp	Opt.	k	Step	Loss (trained)	Loss ($U_k{=}I$)	Δ
150M	4	Muon	256	2,861	3.357	7.528	+4.17
150M	4	AdamW	256	2,861	3.474	6.911	+3.44
150M	8	Muon	256	2,861	3.371	11.588	+8.22
150M	8	AdamW	256	2,861	3.489	11.138	+7.65
500M	4	Muon	256	9,535	3.061	9.043	+5.98
500M	4	AdamW	256	9,535	3.213	8.432	+5.22
500M	8	Muon	256	9,535	3.088	11.342	+8.25
500M	8	AdamW	256	9,535	3.262	10.040	+6.78
1B	4	Muon	256	19,072	3.051	10.023	+6.97
1B	4	AdamW	256	19,072	3.374	8.602	+5.23
1B	8	Muon	256	19,072	3.081	11.108	+8.03
1B	8	AdamW	256	19,072	3.361	11.510	+8.15

D. Keeping Projectors in the Stiefel Manifold

In this section, we demonstrate the necessity of constraining subspace projectors to the Stiefel manifold during training. If the subspace projectors are trained naively, standard optimizer updates cause them to deviate from the manifold, thereby violating the orthonormality constraint. Consequently, the projection loses its isometric properties, allowing the model to learn representations outside the intended subspace.

To empirically validate this, we train a 150M parameter model under three distinct projection configurations. In the baseline setup, we initialize an orthogonal projector and freeze it for the duration of training. In the second configuration, the projector is updated using standard Euclidean

gradients without any manifold constraints. As anticipated, these unconstrained updates cause the weight matrix to escape the manifold and lose its orthogonality, which actively degrades performance—yielding worse results than simply utilizing a fixed projection. Finally, enforcing the manifold constraint via SPEL (Yang & Lai, 2026) updates resolves this issue and achieves the lowest validation loss. These results are summarized in Table 5.

Table 5. Impact of Stiefel manifold constraints on validation loss for a 150M parameter model. Unconstrained updates (Muon) degrade performance compared to a fixed projection, while SPEL updates successfully optimize the projector while maintaining orthogonality.

Optimizer	Val loss	Delta
Fixed orthogonal (No updates)	3.1673	1.19%
Learnable orthogonal (Muon)	3.2101	2.56%
Learnable orthogonal + SPEL (SPEL)	**3.1564**	**0.84%**

E. Rank Collapse in Weight Matrices

Figure 5 illustrates the induced rank collapse observed when co-training the projection matrix on the Stiefel manifold. Learning the projection encourages the model to develop a low-rank structure that aligns closely with the learned basis. Consequently, the model exhibits a significantly more pronounced rank collapse compared to a fixed orthogonal projection. This structural adaptation accounts for the higher percentage of residual energy captured within the subspace.

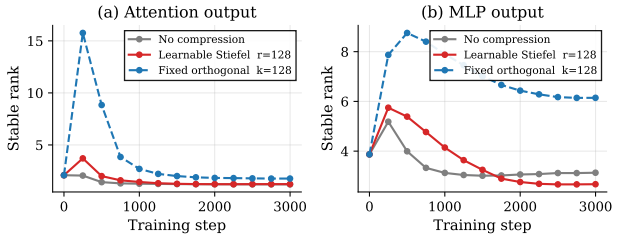

Figure 5. Dynamics of the stable rank, defined as $\mathrm{srank}(A) = \frac{\|A\|_F^2}{\|A\|_2^2}$, for layers adjacent to the compression boundary over the course of training. (a) Under a learnable Stiefel compression ($r = 128$), the attention output projection rapidly collapses to a severely low-rank structure, converging toward near rank-1 behavior. In contrast, a fixed orthogonal compression initially triggers a large transient increase in stable rank before partially collapsing. (b) Similarly, the MLP output projection adapts toward a lower rank under the learned projector, whereas the fixed orthogonal basis maintains a substantially higher stable rank throughout training. Together, these results demonstrate that jointly learning the projection matrix compels the network to reorganize its activations into a compact, low-dimensional subspace aligned with the communication rank. This explains the substantially higher residual energy retention achieved by the learned projector.

F. Implementation Details

In this section, we provide a comprehensive overview of the models, datasets, and hyperparameters used across all experiments. Our goal is to ensure full reproducibility of the pipeline-parallel compression results presented in the main text. We ran our experiments on nodes consisting of 4× Nvidia H100 GPUs with precision bf16, using the PyTorch (Paszke et al., 2019) framework with HuggingFace Transformers (Wolf et al., 2019) implementation of LLaMA 2 (Touvron et al., 2023b), with RoPE (Su et al., 2024) positional embeddings. We used Python 3.10.

F.1. Model Architectures

All experiments evaluate decoder-only transformers based on the LLaMA 2 architecture (Touvron et al., 2023b). We define three scales of models: 150M, 500M, and 1B parameters. To ensure stable comparisons across scales, all models use a context sequence length of 2048 tokens and process inputs using a vocabulary size of 32,000. The structural hyperparameters for each model scale are as follows:

- **150M:** $d_{\text{model}} = 1024$, $n_{\text{layers}} = 9$, $n_{\text{heads}} = 8$, $d_{\text{ffn}} = 2688$
- **500M:** $d_{\text{model}} = 1536$, $n_{\text{layers}} = 18$, $n_{\text{heads}} = 12$, $d_{\text{ffn}} = 4096$
- **1B:** $d_{\text{model}} = 2048$, $n_{\text{layers}} = 20$, $n_{\text{heads}} = 16$, $d_{\text{ffn}} = 5440$

F.2. Dataset and Token Budget

Models were pre-trained on the DCLM dataset (Li et al., 2024). For all runs, we adhered to the Chinchilla compute-optimal scaling regime (Hoffmann et al., 2022), defining our total training steps such that the models process 20 tokens per parameter. For example, the 150M model was trained on roughly 3 billion tokens, the 500M model on 10 billion tokens, and the 1B model on 20 billion tokens. A uniform 5M token split from DCLM was held out entirely for validation.

F.3. Optimizer Configuration

We employed a hybrid optimizer approach. The 2D hidden weight matrices (attention projections and MLP matrices) were updated using the Muon optimizer (Jordan et al., 2024), using PolarExpress (Amsel et al., 2026) algorithm for orthogonalizing the updates. The 1D parameters, including biases, layer normalizations, token embeddings, and the final language modeling head, were updated using AdamW (Loshchilov & Hutter, 2019).

Hyperparameter Sharing Strategy. To ensure a fair comparison between the baseline (uncompressed) models and

our proposed methods and baselines (MDCP-PP, SSN, etc.), we used the exact same optimizer learning rates and weight decay as the uncompressed baselines across all compressed experiments, and we did not tune them specifically for the compression techniques. For the main model parameters, we set the Muon learning rate to $\eta_\mu = 0.02$ and the AdamW learning rate to $\eta_{\text{adam}} = 0.01$ (a $0.5\times$ multiplier on the Muon rate). For the 1B model experiments, we used $\eta_\mu = 0.01$ and $\eta_{\text{adam}} = 0.005$. We use a global batch size of 512 for all runs.

F.4. MDCP-PP Compression Hyperparameters

Our method relies on learning an orthogonal projector $A_p \in \text{St}(d, r)$ per pipeline stage boundary. We optimized these Stiefel manifold projectors using the manifold constrained steepest descent algorithm (Yang & Lai, 2026) via a Polar Express retraction (Amsel et al., 2026). We found empirically that the projector optimization benefits from a slightly reduced learning rate compared to the main model weights; therefore, we set the projector learning rate to be $0.1\times$ the main Muon learning rate.

For the Vector Quantized (MDCP-PP + VQ) experiments, we used a codebook size of $K = 256$ and applied $R = 2$ residual quantization rounds with a group dimension of $G = 2$, utilizing an asynchronous streaming dictionary with streaming interval 5, to amortize codebook synchronization costs across micro-batches.

G. Amortizing VQ: Streaming Codebook Synchronization

For our vector-quantized variant (MDCP-PP + VQ), both the sender and receiver stages must maintain identical codebooks. Transmitting the full codebook $\mathcal{C}_p \in \mathbb{R}^{r\times K}$ across the network at every micro-batch would severely erode the bandwidth savings achieved by quantization.

Motivated by the observation that VQ codebooks evolve slowly over training (Zhang et al., 2024), we introduced an asynchronous streaming dictionary update protocol (§3.5). Instead of full synchronization, the sender transmits only a small, randomly sampled subset of the codebook (e.g., $1/K$ of the codes) alongside the quantized activations in each micro-batch.

To validate that this partial synchronization does not degrade representational fidelity, we conducted an ablation comparing full codebook synchronization against our streaming protocol on a 150M model ($P = 4$, $8\times$ compression). The model trained with **full synchronization** achieved a validation loss of 3.1647, while the model trained with the **streaming dictionary protocol** achieved an essentially identical validation loss of 3.1651. This result confirms that continuous, fractional codebook updates are sufficient to maintain alignment between pipeline stages without incurring the communication penalty of full synchronization.

H. Ablation of Factorized Anchors

In Section 3.2, we hypothesized that the token-driven offset in the residual stream is inherently high-rank and consumes projector capacity if not explicitly modeled. To validate this, we ablate the anchor formulation on a 150M model across an 8-stage pipeline ($P = 8$, $4\times$ compression), a depth where compression errors severely compound.

As shown in Table 6, removing the anchor entirely degrades the validation loss to 3.209 (a $+2.39\%$ degradation relative to the uncompressed baseline). Adding a static, frozen embedding projection (analogous to the static offset in SSN) fails to improve performance (3.212). Conversely, utilizing a fully trainable per-stage anchor recovers performance to 3.149, almost matching the uncompressed baseline (3.134), but incurs an unacceptable parameter overhead (an additional full embedding table per stage).

Our proposed *factorized anchor* strikes the optimal balance: by learning a low-rank trainable component (E_p^{small}) against a frozen random projection (P_p), it recovers the majority of the performance (3.165, only $+1.11\%$ degradation) while adding negligible parameter overhead to the pipeline stages.

Table 6. Ablation of anchor embedding strategies (150M model, $P = 8$, $4\times$ compression). The factorized anchor recovers performance comparable to a fully trainable anchor but with negligible parameter overhead.

Anchor Strategy	**Val Loss**	**Δ %**
Uncompressed Baseline ($P = 8$)	3.134	—
No Anchor	3.209	+2.39%
Decoupled embedding (SSN (Ramasinghe et al., 2025) style)	3.212	+2.48%
Full Trainable Anchor	**3.149**	**+0.47%**
Factorized Anchor (Ours)	3.165	+1.11%

I. Summary of Hyperparameters

Table 7. **Hyperparameters shared across all methods and scales.**

Hyperparameter	**Value**
Global batch size	512
Micro-batch size	4
Sequence length	2048
Pipeline-parallel degree P	$\{4, 8\}$
Precision	bf16
Attention implementation	SDPA
Muon momentum	0.95
Weight decay	0.01
Gradient clipping	off (1.0 for VQ runs at 1B)
Validation split	5M tokens, held-out from DCLM (Li et al., 2024)

Table 8. **Method-specific hyperparameters for the runs in Table 1.** "LR scale" is $\eta_{\text{adam}}/\eta_\mu$, the AdamW-to-Muon learning-rate multiplier on the 1D parameters (embeddings, biases, LayerNorms, LM head). The SSN baselines update the shared subspace U_k via Grassmann manifold steps every 500 optimizer steps with a fixed Grassmann learning rate of 0.01. "Compression" values are reported relative to a 2-byte (bf16) per-channel payload at the corresponding hidden dimension; SSN, MDCP-PP, and MDCP-PP+VQ are all rank-matched at r=256. Identical values across the three model scales are denoted "–".

Method	Hyperparameter	150M	500M	1B
Uncompressed	Optimizer (2D / 1D)	Muon / AdamW	–	–
	LR scale ($\eta_{\text{adam}}/\eta_\mu$)	0.5	–	–
SSN (AdamW)	Subspace rank k	256	–	–
	Grassmann update period / LR	500 steps / 1×10^{-2}	–	–
	Optimizer	AdamW	–	–
	AdamW learning rate	3×10^{-3}	3×10^{-3}	2×10^{-3}
SSN + Muon	Subspace rank k	256	–	–
	Grassmann update period / LR	500 steps / 1×10^{-2}	–	–
	LR scale ($\eta_{\text{adam}}/\eta_\mu$)	0.1	–	–
MDCP-PP	Projection rank r	256	–	–
	Compression ratio (vs. bf16)	4×	6×	8×
	Anchor rank ($E_p^{\text{small}} \in \mathbb{R}^{V\times r}$)	256	–	–
	Anchor projector P_p	frozen, random orthonormal	–	–
	SPEL LR multiplier α/η_μ	0.1	–	–
	SPEL retraction (LMO / projection steps)	5 / 7, Polar Express (Amsel et al., 2026)	–	–
MDCP-PP + VQ	All MDCP-PP hyperparameters	(as above)	–	–
	Codebook size K / residual rounds R	256 / 2	–	–
	VQ group size G	2	–	–
	Streaming dictionary refresh	$1/5$ codes per micro-batch	–	–
	Compression ratio (vs. bf16)	8×	12×	16×