COMPASS: ANCHOR-GUIDED COMMUNICATION FOR DECENTRALIZED LANGUAGE MODEL POST-TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Pipeline-parallel post-training over Internet links must reduce communication while adapting open-weight models *without* architectural changes. The two-circuit design of Ramasinghe et al. (2026) addresses this problem with a fast circuit that uses lossy compression in the forward and backward passes, and an anchor circuit that runs uncompressed passes on delayed weights. The anchor trades staleness for uncompressed computation and uses its gradients to guide the fast circuit. We propose COMPASS, which improves both pipeline compression and communication between the circuits. We propose to learn a basis from the anchor activations, reducing payloads at compressed boundaries by more than two orders of magnitude relative to dense communication in supervised fine-tuning (SFT). Across SFT and reinforcement learning with verifiable rewards (RLVR) experiments, we find that anchor-gradient *signs* are sufficient to provide useful guidance, reducing the cost of returning gradients. For RLVR, we propose a weight forecasting mechanism that accounts for anchor delay as the policy and its training data change. We also derive a convergence bound for the system and show that it achieves state-of-the-art results across various SFT and RLVR benchmarks.

1 INTRODUCTION

Training large language models typically requires tightly connected datacenters beyond most researchers' reach. Pooling GPUs over ordinary Internet links is cheaper and has motivated distributed inference (Borzunov et al., 2023) and pretraining (Ramasinghe et al., 2025b; Aboudib et al., 2026; Janson et al., 2026; Avraham et al., 2026). Full-parameter post-training of open-weight models over these links remains far less studied, particularly reinforcement learning. Communication-efficient pretraining methods use weight subspaces (Ramasinghe et al., 2025b), residual bottlenecks between stages (Aboudib et al., 2026), or learned subspace projections (Janson et al., 2026) from the first training step, making them difficult to adopt for existing models.

We study pipeline-parallel supervised fine-tuning (SFT) and reinforcement learning, where neighboring stages exchange activations and their gradients every step. This pipeline-parallel (PP) traffic becomes the bottleneck over slow links (Narayanan et al., 2021; Ramasinghe et al., 2025b). Lossy compression reduces this traffic, but changes both the forward computation and the gradients used for learning. We aim to reduce communication while retaining useful guidance for optimization and preserving the model architecture.

Ramasinghe et al. (2026) proposed a *two-circuit* design for this tradeoff. The fast circuit uses lossy activation and gradient communication in the forward and backward passes. The anchor runs uncompressed passes but works with delayed weights, trading compression error for staleness. Its stale gradients provide a guidance signal that helps the fast circuit compensate for lossy computation. Fast workers continue training asynchronously, without waiting for the anchor.

COMPASS retains this design and improves communication within and between the circuits. The earlier method (Ramasinghe et al., 2026) uses a random mask, which discards activation coordinates without using their structure. We instead learn a compact basis from the anchor's uncompressed activations. By using this structure, the fast circuit can transmit fewer values while retaining information needed for adaptation. In our supervised setting, this reduces payloads at compressed

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

boundaries by $128\times$ relative to dense communication, compared with $20\times$ for the earlier random mask.

The second improvement concerns what the anchor needs to send back. We find that the signs of its gradient history are sufficient to provide useful guidance in our supervised and reinforcement-learning experiments. Our *Signed-EMA* correction uses these signs to guide the update direction, while the fast circuit supplies magnitudes from its current gradients. This reduces gradient-return communication without sending detailed anchor-gradient values.

Reinforcement learning adds a further difficulty: the model generates its own training data, so both the policy and the data change while anchor weights are delayed. We introduce *weight forecasting* to account for this lag when computing anchor guidance. We evaluate the method on reinforcement learning with verifiable rewards (RLVR), keeping compression active throughout training. We also derive a convergence bound for the supervised sign-based correction under stated assumptions.

2 Related Work

Communication-efficient DP training. Local updates reduce DP synchronization frequency (Stich, 2019; Douillard et al., 2023; Jaghouar et al., 2024; Peng et al., 2024; Sani et al., 2024). Streaming DiLoCo overlaps synchronization with computation; low-rank compressors reduce message sizes (Douillard et al., 2025; Vogels et al., 2019; Wang et al., 2018; Basu et al., 2019; Zhao et al., 2024). We retain the two-circuit baseline's Streaming DiLoCo and PowerSGD, focusing on pipeline and anchor links.

Activation compression. Activation compression changes inter-layer communication (Rudakov et al., 2023; Ramasinghe et al., 2025a). AQ-SGD caches per-example activations and quantizes their changes between epochs (Wang et al., 2022). Stateless TAH-Quant allocates bits by tile and rotates tiles to suppress outliers (He et al., 2025). Weight subspaces (Ramasinghe et al., 2025b;c; Wang et al., 2021), residual bottlenecks between stages (Aboudib et al., 2026), and learned subspace projections (Janson et al., 2026) allow stronger compression but require architectural structures during pretraining that released checkpoints lack. Agora combines pipeline-parallel model sharding with fault-tolerant collectives for permissionless pretraining over Internet links (Avraham et al., 2026). Our direct baseline, Ramasinghe et al. (2026), combines random activation masking with asynchronous gradient guidance (Section 3). Section 4 describes our changes.

RL with verifiable rewards. RLVR and group-relative policy optimization improve reasoning through automatically checked rewards (Lambert et al., 2024; Shao et al., 2024; Yu et al., 2025; Zheng et al., 2025). Systems such as verl and vLLM separate training from rollouts (Sheng et al., 2024; Kwon et al., 2023). The earlier two-circuit study used unmasked GSPO after compressed supervised training. We keep compression active during RLVR and examine whether training stays close to its dense control as the policy determines later rollouts. Sampling and evaluation remain dense.

3 Preliminaries: Two-Circuit Training

COMPASS builds on the topology of Ramasinghe et al. (2026): an $X \times Y$ GPU mesh with X data-parallel replicas of Y pipeline stages. Pipeline neighbors exchange activations and their gradients; data-parallel workers synchronize corresponding shards. A *fast circuit* of $(X-1) \times Y$ workers takes optimizer steps with compressed pipeline messages. An *anchor circuit* of $1 \times Y$ workers holds one complete replica, periodically receives a recent checkpoint, and returns gradients from an unmasked forward and backward pass. The anchor takes no optimizer steps, and fast workers never wait for it. Guidance using weights from step τ and received at step t is $K = t - \tau$ steps old, including checkpoint transfer, the dense pass, and gradient return.

In Ramasinghe et al. (2026), senders retain random activation coordinates at target keep fraction ρ, and receivers rescale them by $1/\rho$. A shared seed key reconstructs the mask without transmitting it or per-entry indices. The mask belongs to the computation graph, so forward activations and backward gradients share the same sparsity pattern and nominal $1/\rho$ payload reduction at each compressed

boundary. We show that replacing this random mask (PRF mask) with a *learned basis* allows higher compression rates.

Fast workers maintain an exponential moving average M_t of arriving anchor gradients, separate from AdamW's momentum. For each targeted weight matrix, the compact singular value decomposition $M_t = U_t S_t V_t^\top$ defines a filter that preserves directions supported by past gradients and suppresses weak ones. With singular values s_i, threshold $\lambda > 0$, and $D_t = \mathrm{diag}(s_i/(s_i + \lambda))$, they filter and blend the current compressed gradient G_t^{fast}:

$$\begin{aligned} G_t^{\mathrm{filt}} &= U_t D_t \big(U_t^\top G_t^{\mathrm{fast}} V_t\big) D_t V_t^\top, \\ G_t^{\mathrm{prior}} &= a G_t^{\mathrm{fast}} + (1-a) G_t^{\mathrm{filt}}, \qquad a \in [0,1]. \end{aligned} \tag{1}$$

AdamW uses G_t^{prior}: the delayed gradient supplies filtering *directions*. The matrices U_t, V_t describe weight-gradient directions. Signed-EMA replaces this filter with the *sign* of the anchor's gradient history and current fast-gradient magnitudes, removing per-matrix SVDs and reducing anchor guidance to one bit per coordinate.

4 METHOD

Using the topology in Section 3, COMPASS makes three changes: a learned activation basis replaces random PP masking, Signed-EMA replaces spectral gradient filtering, and compact messages reduce traffic between fast and anchor circuits. We also study compression and delayed guidance during RLVR, where the policy generates new training data, using an anchor-learned basis with its own payload budget and matched codec controls. Figure 1 summarizes the shared design and the RLVR extension.

4.1 LEARNING AN ACTIVATION BASIS ON THE ANCHOR

For supervised adaptation, we replace the random activation mask of Ramasinghe et al. (2026) by learning which directions to transmit from the anchor's uncompressed activations at each selected boundary. Let $h \in \mathbb{R}^{T \times H}$ contain T hidden states of width H at boundary ℓ. With an orthonormal basis $Q_\ell \in \mathbb{R}^{H \times r}$, the sender computes coefficients and the receiver reconstructs activations as

$$P = hQ_\ell, \qquad \hat{h} = PQ_\ell^\top. \tag{2}$$

The link carries Tr values instead of TH, reducing payload by H/r. Because the projection is part of the training graph, backward derivatives pass through the same r-dimensional space. Both directions therefore receive equal compression at each selected boundary.

The anchor maintains a covariance average with decay β_Q and extracts a rank-r basis by subspace iteration (Vogels et al., 2019), refreshing it every C fast steps. The fast circuit only projects and reconstructs (Appendix D for the update). Between refreshes, the fixed basis removes the same activation directions.

4.2 SIGNED-EMA CORRECTION

The fast circuit provides current gradients distorted by compression while the slow circuit provides uncompressed but delayed gradients. These are complementary, thus, we guide fast updates with the slow gradient history. *Remarkably, we find that its signs alone provide useful guidance*, and pairing them with current fast-gradient magnitudes substantially reduces communication between circuits.

The previous method stores the anchor-gradient average on fast workers to filter current gradients through Eq. (1). We replace this spectral filter with a coordinate-wise sign rule and maintain the average on the anchor in fp32:

$$M^{\mathrm{anc}} \leftarrow \beta_{\mathrm{anc}} M^{\mathrm{anc}} + (1-\beta_{\mathrm{anc}}) G_\tau^{\mathrm{anc}}. \tag{3}$$

Here, β_{anc} determines the decay of the gradient history. The anchor sends the signs of this average, which the fast circuit combines with current gradient magnitudes before AdamW:

$$G_t^{\mathrm{corr}} = \alpha G_t^{\mathrm{fast}} + (1-\alpha)|G_t^{\mathrm{fast}}| \odot \mathrm{sign}(M^{\mathrm{anc}}). \tag{4}$$

The blend $\alpha \in [0, 1]$ controls how much of the current gradient is retained. The anchor-aligned term uses the same magnitudes with signs from dense gradient history. This elementwise correction removes the earlier spectral filter's per-matrix SVD. AdamW then uses its moment estimates to determine the final parameter step.

Eq. (4) needs only signs, so we encode one bit per entry and entropy-code changes between successive fields. The receiver recovers the signs exactly (further savings depend on how often they change).

4.3 Compressing Weight Refreshes

We refresh the anchor's existing checkpoint with sparse updates instead of the full weight shards in Ramasinghe et al. (2026). The fast circuit tracks the anchor's reconstructed weights, adds an error-feedback residual to the weight difference, and selects a fraction k_f of blocks by norm. Selected blocks use int8 values and fp16 row scales; the residual carries unsent information into later refreshes. Appendix D gives the full procedure.

4.4 Extension to RLVR

We test whether the anchor-learned basis and gradient-sign correction extend from supervised adaptation to RLVR with GRPO (Shao et al., 2024). The anchor estimates a basis for transmitting combinations of hidden coordinates from its existing dense activation statistics. The rank-r interface in Eq. (2) sends r coefficients per token in each direction, retaining a fraction r/H of the nominal dense BF16 boundary payload. We compare this interface with PRF masking and TAH-Quant adaptations under shared optimizer and anchor-sign settings. Rollout generation, anchor forward and backward passes, and benchmark evaluation remain dense. The rollout policy and compressed trainer can therefore assign different probabilities.

A dense anchor pass on the first sampled batch initializes Q_ℓ and signs before any optimizer update; compression is active from the first update. The anchor then refreshes activation covariance and gradient history with decays β_Q and β_{anc} at cadence C. The trainer uses the latest signs in Eq. (4) with blend α. Old-policy, current-policy, and frozen-reference evaluations share the current compression basis, held fixed through paired probability and gradient computations. New bases become active after the update.

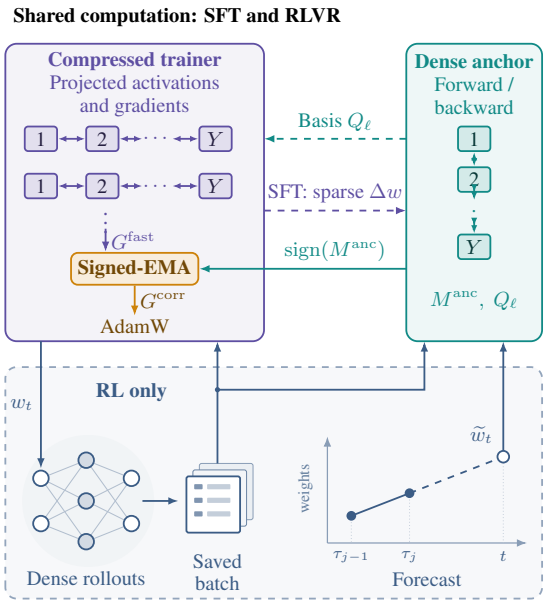

Figure 1: **COMPASS.** The trainer and dense anchor are shared by SFT and RLVR. The anchor sends a basis and gradient-EMA signs; Signed-EMA corrects gradients before AdamW. SFT uses sparse int8 weight refreshes. Blue paths show the RLVR rollout loop and weight forecast. Filled points are delayed exact checkpoints; the open point is the forecast. Forecast anchors reuse current batches; earlier anchors use checkpoint-paired batches. Basis and signs refresh every 20 RL updates; signs are reused between refreshes.

Forecasting anchor weights. The imposed checkpoint delay leaves the anchor at earlier weights. To forecast toward the current update, we use secant extrapolation from two exact checkpoints admitted under this delay, with timestamps $\tau_{j-1} < \tau_j \le t - K$ at trainer update t:

$$\widetilde{w}_t = w_{\tau_j} + \gamma \frac{t - \tau_j}{\tau_j - \tau_{j-1}} \left(w_{\tau_j} - w_{\tau_{j-1}} \right), \tag{5}$$

where γ controls forecast gain. Before two admitted checkpoints are available, the anchor uses the latest admitted exact checkpoint and its paired saved batch; once forecasting is available, it uses the current sampled batch. Both cases reuse saved responses, rewards, and advantages without another rollout or optimizer step.

Anchor objective. At selected weights w, the dense anchor differentiates the response-masked surrogate $-A \log \pi_w$, where A denotes saved advantages, plus the configured reference-KL term. It omits the trainer's PPO ratio and clipping. Reference log probabilities are computed densely at initialization, i.e., periodic anchors reuse those cached with their selected batch. For ordinary training batches, these values are computed with the trainer's active compression basis. The anchor provides activation-covariance statistics and gradient signs. Section 6.4 and Appendix H give the settings and comparison protocol.

4.5 Communication Savings by Component

Table 1 reports each supervised change's communication savings, separating comparisons with Ramasinghe et al. (2026) from further savings within our encoding scheme.

Table 1: Communication savings. Reference/new payload ratios.

Ours	Ramasinghe et al. (2026)	Reduction ↑
Learned basis	Random PP mask	$6.4\times$ per compressed boundary
Signed-EMA	PowerSGD gradient return	≈ 1.6–$2.1\times$
Encode sign changes	Complete one-bit sign field	$\sim 2\times$
Sparse int8 refresh	Full bf16 checkpoint	$20\times$

5 Analysis

For Signed-EMA, we analyze how the fast compressed gradient and anchor sign field contribute to descent in the SGD update $w_{t+1} = w_t - \eta u_t$, where $u_t = \alpha g_t + (1-\alpha)|g_t| \odot s_t$, on a fixed supervised objective. Appendix G gives assumptions and proofs.

Let $\nabla F_t := \nabla F(w_t)$ and $\bar{g}_t = \mathbb{E}[g_t \mid \mathcal{F}_t]$, where $\mathcal{F}_t$ contains current weights and anchor signs available before sampling the current fast minibatch. Direction retention satisfies $\langle \nabla F_t, \bar{g}_t \rangle \geq (1-\kappa)\|\nabla F_t\|^2 - \delta$; magnitude retention satisfies $\sum_i |\nabla F_{t,i}|\,|\bar{g}_{t,i}| \geq (1-\kappa_m)\|\nabla F_t\|^2 - \delta_m$. Here, $\kappa, \kappa_m < 1$ describe relative loss, and $\delta, \delta_m \geq 0$ allow additive error.

Proposition 1. *For any vectors a, b, $\sum_i |a_i||b_i| \geq \langle a, b \rangle$. Thus, a codec preserves alignment at least as well through magnitudes as through directions. Any valid direction-retention constants (κ, δ) are also valid for magnitude retention. The inequality is strict at every coordinate whose sign the codec flips.*

Let $\psi < 1/2$ bound the anchor's sign-error rate, weighted by $|\nabla F_{t,i}|\,\mathbb{E}[|g_{t,i}| \mid \mathcal{F}_t]$, and let $\nu \in [0,1]$ lower-bound the fraction of expected fast-gradient energy where fast and anchor signs disagree. Here, ψ depends on the anchor signs' lag behind the true gradient (Lemma 5), and ν on how often compressed-gradient signs are unreliable.

Theorem 2. *For a fixed $\alpha \in [0,1]$, suppose the history and binary-sign conditions above and Assumptions 1–5 hold. In particular, F is L-smooth and bounded below by $F^\star$, and $\mathbb{E}\|g_t\|^2 \leq G^2$. Then, for any $\eta > 0$ and $T \geq 1$,*

$$\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\|\nabla F(w_t)\|^2 \leq \frac{F(w_0) - F^\star}{\mu_\alpha \eta T} + \frac{\delta_\alpha}{\mu_\alpha} + \frac{L\eta G^2[1 - 4\alpha(1-\alpha)\nu]}{2\mu_\alpha},$$

$$\mu_\alpha = \alpha(1-\kappa) + (1-\alpha)(1-2\psi)(1-\kappa_m) > 0, \qquad \delta_\alpha = \alpha\delta + (1-\alpha)(1-2\psi)\delta_m.$$

The bound has three implications. First, the descent coefficient μ_α is a convex combination of signal retained by the fast gradient's direction, $1-\kappa$, and anchor-signed magnitudes, $(1-2\psi)(1-\kappa_m)$. Anchor signs therefore carry the update when $(1-2\psi)(1-\kappa_m) \geq 1-\kappa$: compression loses more through direction than anchor sign errors lose through magnitudes. By the proposition, magnitudes retain at least as much signal to begin with.

Second, blending damps updates when the sources disagree, reducing the second-moment term by the factor $1 - 4\alpha(1-\alpha)\nu$ (Lemma 4). A partial blend thus takes a smaller step on coordinates with uncertain direction and a full step elsewhere.

Third, sign errors occur only where the current gradient magnitude is smaller than the anchor discrepancy (Lemma 5). This discrepancy is bounded by a weighted sum of checkpoint age, EMA history, reconstruction error, and gradient noise. Weight refreshes and RLVR forecasting affect these

Table 2: SFT performance and system cost. We compare Dense, the native 20× baseline, its 128× control, and COMPASS. Higher scores are better. The lower block reports 20-epoch SQL stress scores and system costs.

Domain	Benchmark	Metric	Dense	Ramasinghe et al. (2026)	Ramasinghe et al. (2026) control	COMPASS (ours)
PP payload reduction ↑			1×	20×	**128**×	**128**×
Medical	MedQA	Accuracy ↑	0.275	**0.290**	0.156	0.285
	MedMCQA	Accuracy ↑	0.325	0.333	0.298	**0.337**
Code	HumanEval	pass@1 ↑	0.170	**0.180**	0.120	**0.180**
	MBPP	pass@1 ↑	0.148	**0.159**	0.122	0.155
SQL	Spider	Exec. match ↑	0.100	**0.140**	0.120	**0.140**
Science	ARC-Challenge	Norm. accuracy ↑	0.374	**0.385**	0.226	0.380
	SciQ	Accuracy ↑	**0.948**	0.946	0.810	0.941
Commonsense	HellaSwag	Norm. accuracy ↑	0.551	0.590	0.431	**0.595**
	WinoGrande	Accuracy ↑	0.541	0.574	0.366	**0.577**
Summarization	CNN/DailyMail	ROUGE-1 ↑	0.176	**0.190**	0.128	0.185
	XSum	ROUGE-1 ↑	0.115	0.122	0.107	**0.124**
Math	GSM8K	Exact match ↑	**0.260**	0.250	0.210	**0.260**
Document parsing	QuALITY	Accuracy ↑	0.279	**0.287**	0.218	0.284
	NarrativeQA	Token F1 ↑	0.558	0.579	0.550	**0.590**
Mean (14 benchmarks) ↑			0.344	0.359	0.276	**0.360**
SQL stress	Spider, 20 ep	Exec. match ↑	**0.84**	0.78	0.73	0.79
System cost	Traffic	MiB/step ↓	1,800	160	**95**	110
	Anchor age	K (fast steps) ↓	–	**150**	170	165
	Throughput	Tokens/s ↑	565	7,200	–	**18,100**
	Time for 100M tokens	Hours ↓	49.18	3.86	–	**1.53**

Table 3: Activation-compression baselines on Llama-3.2-1B. Comp. denotes the forward/backward compression ratio at pipeline boundaries, and HE denotes HumanEval pass@1. Zero scores indicate runs did not converge.

Method	Comp. ↑	MedQA ↑	HE ↑	MBPP ↑	GSM8K ↑
Dense	1×	0.275	0.170	0.148	**0.260**
AC-SGD (Wang et al., 2022)	$8/2$×	0.25	0.00	0.00	0.00
TAH-Quant (He et al., 2025)	$8/2$×	0.25	0.00	0.00	0.00
4-bit quantization	4×	0.25	0.13	0.11	0.18
Top-k sparsification	10×	0.23	0.12	0.09	0.16
Structured sparsification (Ramasinghe et al., 2025a)	10×	0.25	0.13	0.10	0.17
Bottleneck projection	20×	0.26	0.14	0.12	0.12
Ramasinghe et al. (2026)	20×	**0.290**	**0.180**	**0.159**	0.250
COMPASS (ours)	**128**×	0.285	**0.180**	0.155	**0.260**

quantities. For the projection codec with fixed downstream error, Lemma 6 expresses κ in closed form using gradient energy outside the basis.

6 Experiments

6.1 Supervised Adaptation

In the main setting, we fine-tune Llama-3.2-1B on an 8×8 (DP×PP) mesh with seven fast replicas, one anchor, and 200 Mbps links (Appendix C). The *Dense* baseline uses uncompressed PP and compressed DP. The method of Ramasinghe et al. (2026) uses random masking with target keep fraction $\rho = 0.05$ and spectral correction. For COMPASS, we use rank $r = 16$, Signed-EMA blend $\alpha = 0.1$, covariance decay $\beta_Q = 0.90$, and basis update interval $C = 200$ fast steps.

We compare COMPASS with Ramasinghe et al. (2026) on 14 benchmarks (Table 2). Our method performs on par with the previous method at higher compression and substantially outperforms its matched payload control. Table 3 compares other activation-compression methods at their native compression ratios. Two-bit quantization fails on the generative tasks, while sparsification and 4-bit quantization recover only part of the dense model's accuracy.

6.2 End-to-End SFT Efficiency

For the same compressed DP stack and 100M-token budget, we estimate a $2.52\times$ speedup over the previous method and orders of magnitude speed up over the uncompressed dense baseline (Table 2) .

Table 4: Code-domain results across model families. Each entry reports HumanEval / MBPP pass@1 (↑).

Model	Size	Dense	Ramasinghe et al. (2026)	COMPASS
Llama-1B	1.0B	0.170 / 0.148	**0.180 / 0.159**	**0.180** / 0.155
Qwen-1.5B	1.5B	**0.213 / 0.188**	0.206 / 0.177	0.205 / 0.18
Gemma-2B	2.0B	**0.201 / 0.174**	0.194 / 0.162	0.190 / 0.159
Phi-2	2.7B	**0.238 / 0.205**	0.228 / 0.193	0.231 / 0.202
Qwen-3B	3.0B	**0.268 / 0.232**	0.260 / 0.222	0.261 / 0.210
OLMo-7.5B	7.5B	**0.335 / 0.294**	0.324 / 0.280	0.327 / 0.281
Llama-8B	8.0B	**0.384 / 0.338**	0.372 / 0.323	0.381 / 0.334

Table 5: COMPASS across mesh sizes. TPS denotes tokens/s at 200 Mbps. The last column gives the speedup over Ramasinghe et al. (2026) at the same mesh.

Mesh	MedQA ↑	HE ↑	GSM8K ↑	TPS ↑	Speedup ↑
4×4	0.282	0.178	0.257	4,500	1.88×
8×4	0.289	0.186	0.264	10,500	1.88×
4×8	0.279	0.173	0.254	7,800	2.52×
8×8	0.285	0.180	0.260	18,100	2.52×
16×8	**0.292**	**0.188**	**0.267**	38,800	2.52×
8×16	0.276	0.171	0.249	28,300	**3.37×**
16×16	0.282	0.178	0.255	**60,600**	**3.37×**

Table 6: Ablations on rank and α.

Knob	Setting	SQL ↑
Rank r	16 — 128× wire	**0.79**
	8 — 256× wire	0.78
Blend α	0.0 — 100% anchor	0.77
	0.1 — 90% anchor	**0.79**
	0.3 — 70% anchor	0.78
	0.5 — 50% anchor	0.74
	0.7 — 30% anchor	0.69

Table 7: Effect of gradient correction across compression methods. All cells target 128× PP compression. Bold marks the best correction for each codec.

Activation codec	None ↑	Spectral ↑	Signed-EMA ↑
Random mask, 16/2048	0.60	**0.73**	0.72
Learned basis, rank 16	0.66	0.75	**0.79**

Table 8: Anchor-link ablations: SQL, 20 epochs. We adopt sign-change encoding and int8 refresh at $k_f = .10$, and drop reduced anchor batches. Bold compares rows within each block. Half and quarter batches halve and quarter only the dense anchor's compute, with unchanged communication (1×). Reduction factors compare with the replaced dense bf16 flow. Each row changes one setting except the stacked row, which combines sign changes, pre-refinement sparse-bf16 refresh, and a half batch.

Knob	Setting	Payload reduction ↑	SQL ↑
Reference	uncompressed connection (bf16)	1×	0.79
Gradient return	sign-change encoding of the sign field	**∼32×**	**0.81**
Weight refresh	int8, k_f=0.10 (per-row fp16 scale)	**∼20×**	**0.77**
	int8, k_f=0.20	∼10×	**0.77**
	adaptive, bf16	∼10×	**0.77**
	int8 + adaptive	**∼20×**	0.75
	sparse bf16, k_f=0.10	∼10×	0.73
Anchor batch	Half of reference anchor batch	**1×**	**0.72**
	Quarter of reference anchor batch	**1×**	0.71
Stacked	sparse bf16 + sign changes + half batch	combined	0.76

Model scale and mesh size. We further compare code generation across model families (Table 4) and vary the mesh size at 200 Mbps (Table 5). To estimate throughput, we model computation and the slowest link (Appendix B), calibrated to the 8×8 setting in Table 2. At matched meshes, this model predicts a larger speedup over Ramasinghe et al. (2026) as pipeline depth increases, since compression reduces the transfer cost at each boundary. Accuracy remains within 5% of the 8×8 reference. It increases slightly with more fast replicas and decreases slightly with depth, where activations pass through more compressed boundaries.

6.3 Ablations

We first examine the projection rank and correction blend (Table 6). Ranks 8 and 16 perform similarly; we choose 16 to retain more directions. A blend of $\alpha = 0.1$ improves accuracy by two points over anchor-only updates. As the anchor's contribution decreases from 90% to 30%, accuracy gradually falls.

To examine the effect of correction, we compare the learned basis with random masking under no correction, spectral correction, and Signed-EMA (Table 7). The learned basis performs better in all three settings.

Table 8 compares anchor connections and weight refreshes (Appendix E). We also vary anchor delay in Table 9, where SQL accuracy steadily decreases with delay. Since the basis is refreshed every ten anchor cycles, this sweep increases both gradient and basis age.

Table 9: Joint gradient and basis staleness on SQL. We use rank 16 and $\alpha = 0.1$, with basis refresh every ten anchor cycles. Increasing K therefore increases both delays.

K (fast steps)	Basis interval	SQL (20-ep) ↑	Δ vs. K=20 ↑
20	200	**0.79**	—
40	400	0.76	−**3**pp
60	600	0.70	−9pp
80	800	0.67	−12pp

Table 10: RLVR benchmark results (%, ↑). Results are at 200 updates for Qwen2.5-Math-1.5B and 100 for Qwen3-4B-Base. Learned Q combines an adaptive anchor-learned basis with sign correction, using fixed hyperparameters. Rollout generation is dense throughout. We report mean@8 for the competition tasks (AMC, AIME and HMMT) and greedy accuracy for the other tasks, with a 3,072-token response budget for both model suites. MATH500 shares 46 prompts with validation; MMLU-STEM uses 760 examples, and OlympiadBench uses 672 after filtering. We match training horizons and decoding within each model, while the model-specific training response budgets differ. Appendix H provides the protocol.

	Qwen2.5-Math-1.5B			Qwen3-4B-Base		
Benchmark	Base	Dense	Learned Q	Base	Dense	Learned Q
MATH500 (ID)	44.80	**67.00**	66.20	69.00	**74.60**	71.60
GSM8K	34.80	**77.63**	77.48	68.46	**90.98**	89.92
Minerva	2.94	**16.18**	15.81	12.87	**28.31**	23.53
OlympiadBench	18.15	**23.21**	21.73	25.74	**29.76**	25.89
MMLU-STEM subset	10.00	35.92	**45.66**	38.29	**65.66**	56.58
AMC23	30.94	**47.81**	44.69	39.06	**57.50**	48.44
AIME24	8.33	10.42	**12.92**	10.42	**11.25**	8.75
AIME25	4.58	5.00	**7.08**	5.00	**15.00**	9.58
AIME26	6.25	**10.00**	7.92	6.25	**10.83**	5.83
HMMT25	0.00	0.42	**0.83**	**2.50**	**2.50**	0.42
Ten-benchmark mean	16.08	29.36	**30.03**	27.76	**38.64**	34.05
Nine-OOD mean	12.89	25.18	**26.01**	23.18	**34.64**	29.88

6.4 RLVR Evaluation

Setup. We train Qwen2.5-Math-1.5B and Qwen3-4B-Base from pretrained checkpoints on MATH (Yang et al., 2024; 2025; Hendrycks et al., 2021), with one training run per configuration. GRPO uses 128 prompts, eight responses each, and one optimizer update per batch. We select the repeated-sign learned-Q settings on 1.5B using greedy accuracy on 499 validation examples, use them for component/codec comparisons, and transfer them to 4B without retuning. We report checkpoints at 200 and 100 updates, respectively; we selected the latter after inspecting later validation scores. Benchmark scores determine neither the configuration nor these checkpoints. Base is pretrained; Dense uses neither boundary compression nor anchor correction.

RLVR hyperparameters. Learned Q combines an adaptive basis with anchor-sign correction. We set $\alpha = 0.25$, $\beta_{\text{anc}} = 0.25$, $\beta_Q = 0.9$, $C = 20$, $K = 20$, and $\gamma = 1$, and use two subspace iterations. The latest signs are reused at every update without decay. At seven emulated boundaries, we use $r = 77$, $H = 1536$ for 1.5B and $r = 128$, $H = 2560$ for 4B, reducing nominal BF16 payload by approximately 95% in both directions. Before update 1, a dense anchor initializes Q and signs without an optimizer update. Appendix H gives optimizer, GRPO, nominal-budget and TAH-emulation details. Codec ablations retain the optimizer and anchor/sign settings.

Benchmark results. We evaluate on MATH500 and nine OOD benchmarks, weighting them equally (Table 10). At 1.5B, learned Q slightly exceeds Dense in the overall and OOD means. The differences vary across benchmarks, with the largest gain on MMLU-STEM. At 4B, learned Q exceeds Base on the mean but falls below Dense on every benchmark. These comparisons hold for the reported model-specific checkpoints.

Ablations on Qwen2.5-Math-1.5B. We compare codecs under the same optimizer and anchor-sign settings (Table 11(a)). Learned Q gives a higher ten-benchmark mean than PRF (30.03% versus 28.72%), while both TAH-Quant adaptations perform similarly to learned Q. Appendix H explains the nominal budgets and metadata precision used in emulation. With an adaptive basis, removing signs reduces validation accuracy from 63.33% to 50.30% (Table 11(b)). With the same frozen initial

Table 11: RLVR codec and component ablations on Qwen2.5-Math-1.5B. Val@100 and Val@200 denote greedy validation accuracy (%) on the same 499 examples after 100 and 200 updates. The 10-task mean averages the benchmarks in Table 10 equally at 200 updates. Only the stated settings vary. *(a)* Codecs use the same optimizer and anchor/sign settings. Bits are nominal per token, boundary and direction. The TAH-Quant budget assumes 16-bit metadata, whereas emulation uses FP32 metadata (1,464 bits); neither measures network costs (Appendix H). We keep the learned-Q configuration fixed. Appendix I gives results on each benchmark. *(b)* The reference uses an adaptive covariance-EMA basis ($\beta_Q = .9$), repeated signs, $K = 20$ and forecast gain 1. The frozen controls use its initial basis. The last row removes both anchors and signs, so it does not isolate forecasting.

(a) Codec comparison

Codec	Nominal bits ↓	Val@100 ↑	Val@200 ↑	10-task mean ↑
Learned Q	1,232	62.93	63.33	30.03
PRF	1,232	63.33	64.93	28.72
PRF + TAH-Quant	**1,208**	**66.33**	62.53	30.08
Learned Q + TAH-Quant	**1,208**	65.53	**66.93**	**30.12**

(b) Anchor, basis and signs

Basis	Anchor	Repeated signs	Val@200 ↑
Adaptive	Yes	Yes	63.33
Adaptive	Yes	No	50.30
Frozen	Yes	Yes	**66.73**
Frozen	No	No	45.49

Table 12: Sign reuse in RLVR. Qwen2.5-Math-1.5B validation accuracy (%, ↑) on 499 examples at 100 / 200 updates. "–" means untested.

Sign reuse	$\beta_Q = 0$	$\beta_Q = .5$	$\beta_Q = .9$
Once	63.13 / 59.72	60.72 / 56.31	63.93 / 56.11
Annealed	**64.73** / 58.72	58.52 / 59.12	64.33 / 56.11
Repeated	–	–	62.93 / **63.33**

Table 13: SFT initialization for dense GRPO. Scores before / after 100 GRPO steps from a shared Llama-3.2-1B checkpoint and reasoning-SFT procedure. PP factors describe SFT; "–" means no SFT.

SFT method	PP factor ↑	MATH500 ↑	GSM8K ↑
No SFT	–	0.080 / 0.160	0.160 / 0.230
Dense	1×	**0.140** / **0.220**	0.240 / **0.320**
Ramasinghe et al. (2026)			
native mask	20×	0.130 / 0.210	**0.250** / 0.310
COMPASS	**128**×	**0.140** / 0.210	0.240 / **0.320**

basis, retaining anchor/sign updates gives 66.73%, compared with 45.49% without them. These results support sign correction. However, the frozen basis also outperforms the adaptive reference, so these runs do not establish a benefit from continued covariance-based refresh. Appendix I compares forecast gains at matched delays, without a zero-gain control.

Table 12 compares basis memory and sign reuse. At $\beta_Q = .9$, applying the latest signs at every update gives 63.33% validation accuracy at 200 updates, compared with 56.11% for single-use or annealed guidance. Table 12 uses $\alpha = \beta_{\text{anc}} = .25$ and $K = 20$; annealing weights correction by $.75^a$. We selected repeated signs with $\beta_Q = .9$ after validation review. The $\beta_Q = .9$ single-use setting also appears in Appendix Table 16.

6.5 Transfer from Supervised Adaptation to RL

We also evaluate compressed SFT as an initialization for dense RL (Table 13). Starting from the same Llama-3.2-1B checkpoint and MATH solutions (Hendrycks et al., 2021), we compare dense SFT, native masking from Ramasinghe et al. (2026), rank-16 COMPASS without anchor-link compression, and no SFT. Each is followed by 100 dense GRPO steps. Appendix A gives the matched training, evaluation and repeated-seed protocol.

7 Conclusion

COMPASS supports post-training open-weight models over Internet links without changing their architecture. A learned activation basis compresses pipeline traffic, while delayed anchor-gradient signs guide current gradients through Signed-EMA. Sign-change encoding and sparse int8 refreshes further reduce SFT anchor traffic. For RLVR, weight forecasting accounts for delayed checkpoints as the policy and sampled data change. Our SFT approach reduces payloads at pipeline boundaries by $128\times$. RLVR uses approximately 95% smaller nominal payloads, and is on par with uncompressed dense model performance. We also bound convergence for sign-guided SGD on a fixed supervised objective under mild assumptions.

8 AI USE STATEMENT

AI assistants supported code review, literature search, the drafting and checking of proofs, and manuscript preparation. The authors verified all content and are solely responsible for the work's correctness, attribution, and integrity.

REFERENCES

Alan Aboudib, Rodrigo Lopez Portillo A., Kalei Brady, and Steffen Cruz. ResBM: Residual bottleneck models for low-bandwidth pipeline parallelism. *arXiv preprint arXiv:2604.11947*, 2026. URL https://arxiv.org/abs/2604.11947.

Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, pp. 7319–7328, 2021.

Gil Avraham, Violetta Shevchenko, Hadi Mohaghegh Dolatabadi, Karol Pajak, James Snewin, Harry Xi, Rodney O'Donnell, Thalaiyasingam Ajanthan, Sameera Ramasinghe, Chamin Hewa Koneputugodage, Shamane Siriwardhana, and Alexander Long. Agora: Collective and permissionless internet-scale pretraining of large language models. *arXiv preprint arXiv:2607.13332*, 2026. doi: 10.48550/arXiv.2607.13332. URL https://arxiv.org/abs/2607.13332.

Debraj Basu, Deepesh Data, Can Karakus, and Suhas Diggavi. Qsparse-local-SGD: Distributed SGD with quantization, sparsification and local computations. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Alexander Borzunov, Dmitry Baranchuk, Tim Dettmers, Max Ryabinin, Younes Belkada, Artem Chumachenko, Pavel Samygin, and Colin Raffel. Petals: Collaborative inference and fine-tuning of large models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pp. 558–568. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.acl-demo.54. URL https://aclanthology.org/2023.acl-demo.54/.

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.

Arthur Douillard, Yanislav Donchev, Keith Rush, Satyen Kale, Zachary Charles, Zachary Garrett, Gabriel Teston, Dave Lacey, Ross McIlroy, Jiajun Shen, Alexandre Ramé, Arthur Szlam, Marc'Aurelio Ranzato, and Paul Barham. Streaming DiLoCo with overlapping communication: Towards a distributed free lunch. *arXiv preprint arXiv:2501.18512*, 2025.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 2024. URL https://zenodo.org/records/12608602.

Guy Gur-Ari, Daniel A. Roberts, and Ethan Dyer. Gradient descent happens in a tiny subspace. *arXiv preprint arXiv:1812.04754*, 2018.

Guangxin He, Yuan Cao, Yutong He, Tianyi Bai, Kun Yuan, and Binhang Yuan. TAH-Quant: Effective activation quantization in pipeline parallelism over slow network. *arXiv preprint arXiv:2506.01352*, 2025.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *NeurIPS Datasets and Benchmarks Track*, 2021.

Sami Jaghouar, Jack Min Ong, and Johannes Hagemann. OpenDiLoCo: An open-source framework for globally distributed low-communication training. *arXiv preprint arXiv:2407.07852*, 2024.

Paul Janson, Edouard Oyallon, and Eugene Belilovsky. Learned subspace compression for communication-efficient pipeline parallelism. *arXiv preprint arXiv:2606.05484*, 2026. URL https://arxiv.org/abs/2606.05484.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles (SOSP)*, 2023.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. Tülu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024.

Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski. Measuring the intrinsic dimension of objective landscapes. In *International Conference on Learning Representations (ICLR)*, 2018.

Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, Amar Phanishayee, and Matei Zaharia. Efficient large-scale language model training on GPU clusters using Megatron-LM. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, 2021.

Bowen Peng, Lizhang Chen, Baiyu Su, Jeffrey Quesnelle, Diederik P. Kingma, and Qiang Liu. DeMo: Decoupled momentum optimization. *arXiv preprint arXiv:2411.19870*, 2024.

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*, 2025a.

Sameera Ramasinghe, Thalaiyasingam Ajanthan, Gil Avraham, Yan Zuo, and Alexander Long. Subspace networks: Scaling decentralized training with communication-efficient model parallelism. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025b.

Sameera Ramasinghe, Thalaiyasingam Ajanthan, Hadi Mohaghegh Dolatabadi, Gil Avraham, Violetta Shevchenko, Yan Zuo, Chamin P Hewa Koneputugodage, and Alexander Long. Mixtures of subspaces for bandwidth efficient context parallel training. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025c.

Sameera Ramasinghe, Shamane Siriwardhana, Thalaiyasingam Ajanthan, Hadi Mohaghegh Dolatabadi, Chamin P Hewa Koneputugodage, Gil Avraham, Violetta Shevchenko, James Snewin, Karol Pajak, Harry Xi, and Alexander Long. Communication-efficient LLM adaptation over decentralized GPU meshes. *arXiv preprint arXiv:2609.14339*, 2026. doi: 10.48550/arXiv.2609.14339. URL https://arxiv.org/abs/2609.14339.

Mikhail I Rudakov, Aleksandr Nikolaevich Beznosikov, Ya A Kholodov, and Alexander Vladimirovich Gasnikov. Activations and gradients compression for model-parallel training. In *Doklady Mathematics*, volume 108, pp. S272–S281. Springer, 2023.

Lorenzo Sani, Alex Iacob, Zeyu Cao, Royson Lee, Bill Marino, Yan Gao, Dongqi Cai, Zexi Li, Wanru Zhao, Xinchi Qiu, and Nicholas D. Lane. Photon: Federated LLM pre-training. *arXiv preprint arXiv:2411.02908*, 2024.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. HybridFlow: A flexible and efficient RLHF framework. *arXiv preprint arXiv:2409.19256*, 2024.

Sebastian U. Stich. Local SGD converges fast and communicates little. In *International Conference on Learning Representations (ICLR)*, 2019.

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

Hongyi Wang, Scott Sievert, Shengchao Liu, Zachary Charles, Dimitris Papailiopoulos, and Stephen Wright. ATOMO: Communication-efficient learning via atomic sparsification. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

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.

Jue Wang, Binhang Yuan, Luka Rimanic, Yongjun He, Tri Dao, Beidi Chen, Christopher Ré, and Ce Zhang. Fine-tuning language models over slow networks using activation quantization with guarantees. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Tianle Wang, Jiayu Liu, Zhongyuan Wu, Shenghao Jin, Wei Chen, Hao Xu, and Ning Miao. Linear dynamics in the RLVR training of large language models. *arXiv preprint arXiv:2601.04537*, 2026.

Zhepei Wei, Xinyu Zhu, Wei-Lin Chen, Chengsong Huang, Jiaxin Huang, and Yu Meng. You only need minimal RLVR training: Extrapolating LLMs via rank-1 trajectories. *arXiv preprint arXiv:2605.21468*, 2026.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, et al. DAPO: An open-source LLM reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

Yi Yu, Tengyao Wang, and Richard J. Samworth. A useful variant of the Davis–Kahan theorem for statisticians. *Biometrika*, 102(2):315–323, 2015.

Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. GaLore: Memory-efficient LLM training by gradient low-rank projection. In *International Conference on Machine Learning (ICML)*, 2024.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*, 2025.

A SFT-TO-RL PROTOCOL

All SFT arms share the same token budget, data order, optimizer and checkpoint rule. We use their terminal checkpoints and retain the unchanged pretrained model as the no-SFT control. Each downstream arm will use MATH training prompts, 128 prompts and eight responses per prompt, a 2048-token response budget, and the same verifier, optimizer, hardware and sampler configuration. The responses will differ as the policies change. Compression and anchor correction will be disabled to isolate the SFT initialization.

Before and after GRPO, we evaluate MATH500 and GSM8K with the same dense greedy harness. Evaluation examples are excluded from training and checkpoint selection. We repeat the comparison across training seeds and report means and standard deviations.

B EXPERIMENTAL PROTOCOLS

Per-domain supervised comparison. We trained a separate model for each domain from the same Llama-3.2-1B checkpoint, matching the training examples and order, token budget, optimizer, global batch, compressed boundaries, and DP stack within that domain. We used the training portions of Spider, GSM8K, and NarrativeQA, with a separate code-instruction corpus for HumanEval evaluation. Evaluation examples and near duplicates were excluded from training; HumanEval prompts, solutions, and tests were used only for evaluation. We selected settings using validation data held out from training, including for GSM8K, and reserved the final evaluation set for scoring. The dataset versions, splits, code corpus, NarrativeQA context format, prompts, and generation limits were fixed before the comparison. We evaluated all models densely with the same decoding and task scorer and computed the mean and standard deviation over at least three paired training seeds at the final token budget.

System-cost protocol. The system-cost rows of Table 2 were profiled on the 20-epoch SQL stress run. All comparisons used the same 200 Mbps links and a total of 8×8 workers. Dense training used all workers for training; the other methods allocated 7×8 to the fast circuit and 1×8 to the anchor. Microbatching and accumulation were adjusted to keep the global batch fixed. We measured fast-training tokens per wall-clock second, excluding anchor replay tokens, and summed bytes sent on PP, DP, and both anchor directions, including basis updates, metadata, and scales. We recorded the median checkpoint age when its guidance reached the fast trainer.

Matched timing protocol. The timing comparison used the same Llama-3.2-1B initialization, SQL data order, global batch, sequence packing, optimizer, and 200 Mbps links. The total allocation remained at 64 workers: eight training replicas for Dense, or seven fast replicas and one anchor for the other methods, with eight pipeline stages per replica. We fixed the DP synchronization schedule and matched the anchor batch and scheduling policy between the two anchored methods. Throughput counted only non-padding fast-training tokens. Timing measurements included initialization, encoding, basis construction, checkpoint transfers, validation, and synchronization, with at least three paired seeds.

Arm configurations. For the random-mask control, we used the mask from Section 3 with exactly 16 of 2048 coordinates per token and inverse-keep scaling of 128. Its payload therefore matches the rank-16 basis coefficient for coefficient. Both stages regenerated the mask from the same key. This is a new control, not an operating point reported in the earlier paper. In both columns based on Ramasinghe et al. (2026), we retained spectral correction, a PowerSGD-compressed anchor return, and full checkpoint refresh. COMPASS uses the learned basis and Signed-EMA with sign-change encoding and top-10% int8 refresh under error feedback. We ablated the two anchor-link compressors separately in Table 8.

Codec and correction swaps. To compare correction rules, we fixed boundary placement, training data, DP synchronization, anchor batches, and checkpoint inputs, with checkpoint delay $K = 20$ and basis refresh every 200 fast steps. Full checkpoint refreshes, dense gradient returns for spectral correction, and full sign fields for Signed-EMA were used, with sign-change encoding and sparse refresh disabled. This comparison excluded the predecessor's return compressor. We preselected each correction's settings with the same validation budget, and the anchor pass schedule remained

active in all six cells. Without gradient correction, the fast gradient entered AdamW unchanged and the anchor gradients were discarded; only the learned-basis row used the anchor's activation statistics. We computed the mean and standard deviation over at least three training seeds using a fixed checkpoint rule, recording final-checkpoint scores separately whenever the rule selected an earlier checkpoint.

C Supervised Training Details

The supervised experiments use Llama-3.2-1B, with approximately 1.23B parameters, 16 transformer layers, hidden size 2048, and 32 attention heads. We use bf16 weights and activations with FlashAttention-2, and AdamW with $(\beta_1, \beta_2) = (0.9, 0.999)$, a peak learning rate of 2×10^{-5}, a cosine schedule with warmup, and gradient clipping. Signed-EMA correction is applied before the gradient enters AdamW, and the anchor momentum is stored in fp32.

The mesh contains a 7×8 fast circuit and a 1×8 anchor circuit, with dimensions ordered as DP $\times$ PP and links throttled to 200 Mbps. The supervised configuration uses slow-basis projection at alternating pipeline boundaries, with rank 16, covariance EMA $\beta_Q = 0.90$, and refresh interval $C = 200$ fast steps. Thus, $128\times$ describes the reduction at a compressed boundary; it is not a measured reduction in total traffic across all boundaries. For DP, we use Streaming DiLoCo with PowerSGD.

We follow the adaptation setting of Ramasinghe et al. (2026), which supplies the fast/anchor architecture, nonblocking unmasked anchor pass, random in-graph masking, anchor-gradient average, spectral correction, compressed anchor-gradient return, and Streaming DiLoCo–PowerSGD stack. In COMPASS, we replace the supervised activation codec and correction rule, move the gradient average to the anchor side, and test sparse int8 anchor refreshes. We also keep communication compression active during RLVR optimization. The earlier work instead used ordinary unmasked DPO and GSPO after comparing dense and compressed supervised starting points. Since Signed-EMA differs from the earlier spectral filter, the earlier convergence analysis does not establish convergence for the new correction.

The available per-domain metrics are NarrativeQA token F1, code pass@1, and Spider execution match. We use Spider-style SQL for 20 epochs in the longer SQL stress run. For generative evaluations, we use the lm-evaluation-harness where applicable (Gao et al., 2024), and task-specific scorers for execution match and token F1. Table 2 lists the eight domain groups and benchmarks from the earlier manuscript alongside the available COMPASS scores. The additional benchmark entries are synthetic draft placeholders, with matched measurements still pending. We compute the mean by weighting all 14 displayed benchmarks equally, rather than weighting the eight domain groups equally. The earlier unspecified Code result was 0.475 for Dense, 0.510 for the previous method, and 0.497 for COMPASS.

D Supervised Training Procedure

The anchor sees dense activations and uses them to update a covariance moving average,

$$C_\ell \leftarrow \beta_Q C_\ell + (1 - \beta_Q) h^\top h. \tag{6}$$

We extract a rank-r basis using PowerSGD-style subspace iteration (Vogels et al., 2019), refresh it every C fast steps, and keep it fixed between refreshes, with the settings in Section 6.1. Basis estimation runs on the anchor, leaving only projection and reconstruction to the fast circuit.

Algorithm 1 shows the work performed on each circuit. We initialize the activation bases and correction state with the first dense anchor pass. The fast circuit then uses the most recent available basis and sign field, which the anchor updates asynchronously. We estimate the bases from dense activation statistics and the gradient EMA from dense gradients. These are separate estimates and need not be refreshed on every fast step.

The sign-change stream preserves the sign-based interface exactly when the sender and receiver start with the same sign field and apply the same ordered changes. Only the anchor stores the full gradient EMA. The reported $32\times$ reduction combines the one-bit sign representation with the

Algorithm 1 Conceptual composition of the supervised COMPASS interfaces

Require: pretrained weights w_0, projection rank r, blend α, EMA coefficients $\beta_Q, \beta_{\text{anc}}$, basis interval C, block fraction k_f

Allocate $(X-1) \times Y$ fast workers and $1 \times Y$ anchor workers
Initialize bases Q_ℓ and the sign field from the first anchor pass
Fast circuit, each optimizer step:
Forward and backward with coefficients hQ_ℓ at compressed boundaries
Synchronize DP updates with Streaming DiLoCo and PowerSGD
Apply any received sign changes to the local sign field
Form G^{corr} using Eq. (4); take an AdamW step
Anchor circuit, each asynchronous cycle:
Receive the sparse int8 weight delta and update the reconstructed weights
Run a dense forward and backward pass to obtain G^{anc}
Update the fp32 gradient EMA by Eq. (3)
Send entropy-coded changes in $\text{sign}(M^{\text{anc}})$
Update each activation covariance by Eq. (6)
When the basis refresh is due, run subspace iteration and send Q_ℓ

observed compressibility of changes between successive fields. The latter depends on how often signs change.

For weight refresh, the sender tracks the receiver's reconstructed view. It adds the accumulated residual to the current delta, selects blocks by norm, and transmits their quantized values and scales. Error feedback retains what the receiver has not yet received. This reduces traffic, but introduces a temporary error in the weights used for dense anchor computation.

E Supervised Ablations

E.1 Projection Rank, Sign Blend, and Basis Refresh

We use 20 epochs for the SQL stress run and compare projection ranks and blends in Table 6. Rank 8 scores 0.78 at a nominal $256\times$ boundary reduction, while rank 16 scores 0.79 at $128\times$. These results are consistent with a low-dimensional adaptation signal (Aghajanyan et al., 2021; Li et al., 2018; Gur-Ari et al., 2018). However, two ranks on one task do not establish the full spectrum or a general rank threshold.

For slow-basis training, the blend sweep favors a small blend, $\alpha = 0.1$. The anchor-only rule $\alpha = 0$ is two points behind, and larger blends reduce the score. The earlier masking system used a larger blend. This is consistent with the possibility that a random mask retains more useful direction information than a fixed projection, giving the fast gradient a larger role in that setting. This interpretation is specific to the measured configurations. In the reported basis-refresh comparison, we found no measurable sensitivity between $C = 100$, 200, and 400 fast steps. The covariance EMA remains $\beta_Q = 0.90$.

E.2 Compressing the Anchor Connection

In Table 8, we compare changes to the anchor connection against the SQL score of 0.79 obtained with an uncompressed return and weight refresh. Sending signs as entropy-coded changes gives 0.81. Since the encoding preserves the sign field, this difference is not an algorithmic benefit of the encoding. The available results do not quantify its run-to-run variation.

Sparse bf16 weight deltas reduce the score to 0.73. When we instead reduce the anchor batch, the score falls to 0.72 at half size and 0.71 at quarter size. The stacked variant in the table includes the half-batch change and differs from the final int8 configuration. These comparisons show that reducing the data used by the anchor to lower its compute can cost more quality than reducing transmission precision.

Table 8 reports best-checkpoint scores. We also give the last-checkpoint scores here because they differ in the weight-refresh sweep. At $k_f = 0.10$, int8 halves the payload of sparse bf16 and improves

the best score from 0.73 to 0.77, but the last checkpoint scores 0.73. Keeping twice as many int8 blocks gives the same best score of 0.77 and a last-checkpoint score of 0.76. Adaptive widening also reaches 0.77 and ends at 0.73; combining it with int8 gives 0.75 and ends at 0.74.

We use per-row fp16 scales for int8 values. The adaptive rule widens the kept fraction from 0.10 toward 0.50 when the residual-to-weight ratio exceeds 5%. This ratio is unrelated to the checkpoint-window length used in the RLVR forecast. The similar best scores suggest a remaining error-feedback limitation, but do not isolate residual lag as its cause. We favor $k_f = 0.10$ when lower traffic matters more and $k_f = 0.20$ when final-checkpoint stability matters more.

E.3 Joint Gradient and Basis Staleness

In Table 9, we vary the delay K and keep the basis refresh at ten anchor cycles. The basis interval therefore grows from 200 to 800 fast steps across the sweep. The score falls steadily from 0.79 to 0.67, without a sharp instability. To separate the effects of stale gradient correction and stale activation projection, we would need a sweep with a fixed basis interval.

We measure delays of $K = 20$–80. We do not measure $K = 174$, so its loss cannot be read directly from the table. Similarly, $K = 35$ lies between the measured points; this sweep does not directly measure quality at that proposed operating point.

F Staleness

The two circuits maintain different states. The fast circuit holds the current trainable weights, local optimizer state, and received bases and signs. The anchor holds dense activation statistics and the fp32 gradient average. By moving basis estimation and gradient-history storage away from the fast step, we place this work on the circuit that already evaluates dense activations. This does not remove its compute or memory cost. The supervised interface returns a compact basis and sign field, and the fast circuit continues to use the previous signals between anchor updates. Their age therefore matters even when communication never blocks a fast step.

Bandwidth and signal age. The two-circuit baseline reports an anchor delay of roughly 20–25 fast steps with $20\times$ activation masking and an already-compressed gradient return. In a communication-dominated regime, more aggressive boundary compression can shorten the fast step, so the same wall-clock anchor cycle spans more optimizer updates. A serial timing model writes this age as

$$K = \frac{T_{\text{pull}} + T_{\text{clean}} + T_{\text{push}}}{T_{\text{fast}}}, \tag{7}$$

where the numerator contains weight refresh, the unmasked pass, and gradient return. Weight deltas and sign changes reduce two terms but not the unmasked pass, and the result depends on scheduling and overlap. A controlled SQL sweep falls from 0.79 at $K = 20$ to 0.67 at $K = 80$, but basis age grows with gradient age. Table 9 and Appendix F give the full sweep and timing assumptions.

The accounting uses a 1.2B-parameter bf16 model, eight pipeline stages, links of 200 Mbps (about 25 MB/s), sequence length 2048, and hidden size 2048. With 150M parameters per stage, a dense bf16 weight shard is 300 MB and takes approximately 12 s to transfer. The reference calculation assigns another 12 s to a dense gradient return and 72 s to the anchor's uncompressed forward and backward traffic. Its fast-step estimate is about 0.55–0.6 s in the aggressive PP-compression regime.

Under the serial model of Eq. (7), those inputs give

$$K_{\text{dense link}} \approx \frac{12 + 72 + 12}{0.55} = 174.5, \tag{8}$$

$$K_{\text{compressed link}} \approx \frac{0.6 + 72 + 0.4}{0.55} = 132.7. \tag{9}$$

The second row reduces the weight pull by $20\times$ and the gradient return by about $32\times$, leaving the dense pass unchanged. Even a cycle dominated entirely by that 72 s pass corresponds to roughly 131 fast steps. Thus these inputs alone do not establish a reduction to $K \approx 35$.

To reach the proposed $K \approx 35$ operating point after overlap, we would need an effective anchor cycle of about 19.25 s at a 0.55 s fast step. This requires a separate timing or scheduling derivation.

Hiding the anchor pass behind fast training removes blocking from the fast circuit, but does not by itself reduce the age of the weights used by the anchor. We therefore retain 35 as an unverified operating-point estimate, instead of a measured result or a consequence of the serial calculation.

The reference configurations also differ. Appendix H of Ramasinghe et al. (2026) already models a compressed anchor gradient return of roughly 1–2 s, whereas the COMPASS comparison above assumes a dense return of 12 s. Thus, the $32\times$ sign-return factor is relative to bf16 transmission; it is not a demonstrated $32\times$ improvement over that earlier compressed path. Compressing each flow is still useful, but an end-to-end comparison requires a consistent baseline, pipeline schedule, and measured cycle time.

G Convergence Analysis of Signed-EMA

We analyze a plain corrected-gradient step on a fixed objective. The theorem identifies sufficient conditions on the signal retained by the fast gradient and the accuracy of the available anchor signs. It shows how the blend trades these against the size of the update. The retention constants still need to be established for the chosen codec and network; below, we examine them for the projection codec with fixed downstream error. The analysis does not cover AdamW or the complete distributed procedure. The spectral-filter analysis of Ramasinghe et al. (2026) has the same scope limitations, but studies a different correction.

G.1 Setup

Let $F(w) = \mathbb{E}_\xi[\ell(w;\xi)]$ be a fixed objective on $\mathbb{R}^d$, let w_0 be deterministic, and write $f_t := \nabla F(w_t)$. We collect w_t, past randomness, and all guidance admitted before sampling the current fast minibatch in the history $\mathcal{F}_t$. The available basis and binary anchor-sign field $s_t \in \{-1,+1\}^d$ are $\mathcal{F}_t$-measurable. We use $\operatorname{sign}(0) = +1$ in this analysis. Any other fixed binary rule at zero gives the same conclusions, whereas a ternary sign operator requires a separate treatment of zero entries.

We write $g_t = G_t^{\mathrm{fast}}$, $\bar g_t = \mathbb{E}[g_t \mid \mathcal{F}_t]$, and $m_{t,i} = \mathbb{E}[|g_{t,i}| \mid \mathcal{F}_t]$. By Jensen's inequality, $m_{t,i} \ge |\bar g_{t,i}|$. Compression may bias the conditional mean, while $g_t - \bar g_t$ includes randomness from the minibatch and any fresh codec randomness. For fixed $\alpha \in [0,1]$ and $\eta > 0$, we set

$$u_t = \alpha g_t + (1-\alpha)|g_t| \odot s_t, \qquad w_{t+1} = w_t - \eta u_t.$$

The measurability requirement is important here. Guidance computed from the current fast batch cannot be held fixed when averaging over the same batch; such a schedule needs a joint-noise analysis.

G.2 Assumptions

Assumption 1 (Smoothness)**.** F has an L-Lipschitz gradient and is bounded below by a finite $F^\star$.

Assumption 2 (Bounded fast-gradient second moment)**.** $\mathbb{E}\|g_t\|^2 \le G^2 < \infty$ for $0 \le t < T$.

Assumption 3 (Direction retention)**.** There are constants $\kappa \in [0,1)$ and $\delta \ge 0$ such that, almost surely at every step, $\langle f_t, \bar g_t\rangle \ge (1-\kappa)\|f_t\|^2 - \delta$.

Assumption 4 (Magnitude retention)**.** There are constants $\kappa_m \in [0,1)$ and $\delta_m \ge 0$ such that, almost surely at every step, $\sum_i |f_{t,i}|\,|\bar g_{t,i}| \ge (1-\kappa_m)\|f_t\|^2 - \delta_m$.

We require these constants to hold uniformly over the stated horizon. Direction retention lower-bounds alignment without assuming an unbiased compressed gradient. Magnitude retention is a weaker condition: by Lemma 3, the same pair of constants suffices. Any strict improvement in the constants must be shown separately.

Assumption 5 (Weighted sign fidelity)**.** Define $e_{t,i} = \mathbf{1}[s_{t,i} \ne \operatorname{sign}(f_{t,i})]$ and $Z_t = \sum_i |f_{t,i}| m_{t,i}$. Set

$$\psi_t = \frac{\sum_i |f_{t,i}| m_{t,i} e_{t,i}}{Z_t} \quad \text{if } Z_t > 0, \qquad \psi_t = 0 \quad \text{if } Z_t = 0.$$

There is a constant $\psi \in [0, 1/2)$ with $\psi_t \le \psi$ almost surely at every step.

This assumption weights sign errors by their contribution to alignment, rather than counting the fraction of correct signs. A small expected anchor discrepancy is insufficient, particularly near a stationary point. Lemma 5 explains the additional relative-error requirement.

G.3 LEMMAS

Lemma 3 (Magnitude alignment). *For any vectors a, b, $\sum_i |a_i||b_i| \geq \langle a, b\rangle$. Consequently, Assumption 3 implies Assumption 4 with the same constants $\kappa_m = \kappa$ and $\delta_m = \delta$.*

Proof. Since $|a_i||b_i| \geq a_i b_i$ for each coordinate, we obtain the claim by summing and applying the direction-retention inequality. The vector inequality is strict exactly when $a_i b_i < 0$ for at least one coordinate. However, strictness at individual points does not necessarily improve the retention constants uniformly along a trajectory. □

Lemma 4 (Agreement damping). *Let $a_{t,i} = \mathbf{1}[s_{t,i} = \operatorname{sign}(g_{t,i})]$. Then $u_{t,i} = g_{t,i}$ on agreeing coordinates and $u_{t,i} = (2\alpha - 1)g_{t,i}$ on disagreeing coordinates. Thus*

$$\|u_t\|^2 = \|g_t\|^2 - 4\alpha(1-\alpha)\sum_i (1 - a_{t,i})g_{t,i}^2 \leq \|g_t\|^2.$$

Equality holds exactly when $\alpha \in \{0, 1\}$ or all nonzero fast-gradient coordinates agree with the anchor.

Proof. When the signs agree, $|g_{t,i}|s_{t,i} = g_{t,i}$; when they disagree, the term is $-g_{t,i}$. Both identities also hold for $g_{t,i} = 0$. Substituting into u_t, squaring, and summing gives the result, since $1 - (2\alpha - 1)^2 = 4\alpha(1-\alpha)$. □

When $\mathbb{E}\|g_t\|^2 > 0$, define

$$\nu_t = \frac{\mathbb{E}[\sum_i (1 - a_{t,i})g_{t,i}^2]}{\mathbb{E}\|g_t\|^2};$$

otherwise set $\nu_t = 0$. Choose any $\nu \in [0, \min_{0 \leq t < T} \nu_t]$. In particular, $\nu = 0$ is always valid. Lemma 4 gives

$$\mathbb{E}\|u_t\|^2 \leq G^2[1 - 4\alpha(1-\alpha)\nu]. \tag{10}$$

Here, ν_t measures disagreement with the noisy fast gradient, while ψ_t measures disagreement with f_t. Neither quantity determines the other. Eq. (10) bounds a second moment, not only noise variance.

Lemma 5 (Sign errors and anchor discrepancy). *Let $D_t = M_t^{\mathrm{anc}} - f_t$ and $s_t = \operatorname{sign}(M_t^{\mathrm{anc}})$. If $e_{t,i} = 1$, then $|f_{t,i}| \leq |D_{t,i}|$. For $Z_t > 0$,*

$$\psi_t \leq \min\left\{1, \frac{\sum_i |D_{t,i}| m_{t,i}}{Z_t}\right\} \leq \min\left\{1, \frac{\|D_t\|\,\|m_t\|}{Z_t}\right\}.$$

For $Z_t = 0$, $\psi_t = 0$ by definition.

More precisely, suppose n anchor updates have been incorporated, with decay $\beta \in [0, 1)$, initialization M_0, and gradients $a_j = \nabla F(\hat{w}_j) + \zeta_j$. Then

$$M_t^{\mathrm{anc}} = \beta^n M_0 + (1-\beta)\sum_{j=1}^{n} \beta^{n-j} a_j,$$

$$\|D_t\| \leq \beta^n \|M_0 - f_t\| + (1-\beta)\sum_{j=1}^{n} \beta^{n-j}\big(L\|\hat{w}_j - w_t\| + \|\zeta_j\|\big).$$

The empty-sum case $n = 0$ is included with $\beta^0 = 1$.

Proof. If the signs disagree, either $f_{t,i} = 0$ or $M_{t,i}^{\mathrm{anc}}$ lies across zero, including at zero itself. Hence, $|f_{t,i}|e_{t,i} \leq |D_{t,i}|$. We multiply by $m_{t,i} \geq 0$, sum over coordinates, and apply Cauchy–Schwarz, together with $\psi_t \leq 1$. Iterating the EMA recursion gives the displayed expansion exactly. Using $\beta^n + (1-\beta)\sum_{j=1}^{n} \beta^{n-j} = 1$, we subtract f_t and apply the triangle inequality and gradient Lipschitzness to obtain the discrepancy bound. □

The EMA bound accounts for initialization and the age of every contributing gradient. It also holds in expectation without an independence assumption. Replacing the noise sum with a constant requires a bound on the expected weighted sum. Marginal noise bounds alone may be insufficient if admission counts or weights depend on the noise. For deterministic weights and $\mathbb{E}\|\zeta_j\| \le \sigma_a$, this contribution is at most $(1-\beta^n)\sigma_a$. These expected-norm bounds still do not establish almost-sure sign fidelity. To see this, consider one dimension with $f_t = \epsilon > 0$, $M_t^{\mathrm{anc}} = -\epsilon$, and $m_t > 0$. Then $\psi_t = 1$ even as $\|D_t\| = 2\epsilon \to 0$.

Lemma 6 (Projection identity with fixed downstream error)**.** *Consider one linear layer with input h, orthogonal projector $P = QQ^\top$, and dense backpropagated error B. Let $\Gamma = h^\top B$. Holding both h and B at their dense values, the projected-input gradient is $\widetilde{\Gamma} = (hP)^\top B = P\Gamma$, and*

$$\langle \Gamma, \widetilde{\Gamma} \rangle = \|Q^\top \Gamma\|_F^2, \qquad \kappa_W := \frac{\|(I-P)\Gamma\|_F^2}{\|\Gamma\|_F^2} \in [0,1] \quad (\Gamma \neq 0).$$

Thus $\langle \Gamma, \widetilde{\Gamma} \rangle = (1-\kappa_W)\|\Gamma\|_F^2$. For $\Gamma = 0$ both sides vanish without defining the ratio. With an unbiased activation mask and the same fixed B, the expected masked-input gradient instead equals Γ.

Proof. Since $P^\top = P = P^2$, we have $(hP)^\top B = P\Gamma$ and $\langle \Gamma, P\Gamma \rangle = \mathrm{tr}(\Gamma^\top P \Gamma) = \|Q^\top \Gamma\|_F^2$. By orthogonality, $\|\Gamma\|_F^2 = \|Q^\top \Gamma\|_F^2 + \|(I-P)\Gamma\|_F^2$. For a mask with $\mathbb{E}[\hat{h}] = h$ and fixed B, linearity gives $\mathbb{E}[\hat{h}^\top B] = \Gamma$. □

This identity holds with a fixed downstream error. It does not establish Assumption 3 for the compressed network. Even in this setting, κ_W may equal 1, whereas the assumption requires a uniform constant below 1 or a compensating additive error. If we allow the downstream error to change to $\widehat{B}$ in the single-boundary model, we obtain

$$\widehat{\Gamma} = P\Gamma + R, \quad R = Ph^\top(\widehat{B} - B), \qquad \langle \Gamma, \widehat{\Gamma} \rangle \ge \|Q^\top \Gamma\|_F^2 - \|\Gamma\|_F \|R\|_F.$$

Other compressed boundaries may also change h, so these additional terms must be controlled to derive global retention constants. In particular, unbiased activations need not give unbiased gradients. For the loss $\frac{1}{2}(w\hat{h} - y)^2$, consider $h = w = 1$, $y = 3/2$, and $\hat{h} = 2z$ with $z \sim \mathrm{Bernoulli}(1/2)$. The dense weight gradient is $-1/2$, whereas the expected masked gradient is $+1/2$.

Activation variance is also insufficient to control gradient retention. Consider $h = \mathrm{diag}(A, \epsilon)$, $A > \epsilon > 0$, and $B = (0,1)^\top$. The leading activation direction is $Q = (1,0)^\top$, but the gradient $\Gamma = (0,\epsilon)^\top$ is lost entirely. Studies of low-dimensional parameter updates and Hessian-gradient alignment (Aghajanyan et al., 2021; Gur-Ari et al., 2018) motivate the use of subspaces. They do not, however, establish alignment with the leading activation-covariance eigenspace.

To study eigenspace stability, we also need an eigengap. For symmetric covariances C and $\widehat{C}$, let $Q_\star$ and $\widehat{Q}_\star$ be the exact leading rank-r eigenspaces, with $r < H$ and $\gamma = \lambda_r(C) - \lambda_{r+1}(C) > 0$. The Davis–Kahan variant of Yu et al. (2015) gives

$$\|\sin\Theta(\widehat{Q}_\star, Q_\star)\|_F \le \frac{2\min\{\sqrt{r}\,\|\widehat{C} - C\|_2, \|\widehat{C} - C\|_F\}}{\gamma}.$$

This bound applies to ideal eigenspaces. For the implemented basis, we would also need bounds on covariance estimation, delay, and the error from finite subspace iteration. We do not measure these errors here, and a refresh-interval sweep does not establish the required conditions.

G.4 Main result

Theorem 7 (Convergence of Signed-EMA, plain step)**.** *In the setup above, suppose Assumptions 1–5 hold. Define*

$$\mu_\alpha = \alpha(1-\kappa) + (1-\alpha)(1-2\psi)(1-\kappa_m), \qquad \delta_\alpha = \alpha\delta + (1-\alpha)(1-2\psi)\delta_m.$$

For any fixed $\alpha \in [0,1]$, $\eta > 0$, and integer $T \ge 1$, $\mu_\alpha > 0$ and

$$\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\|\nabla F(w_t)\|^2 \le \frac{F(w_0) - F^\star}{\mu_\alpha \eta T} + \frac{\delta_\alpha}{\mu_\alpha} + \frac{L\eta G^2[1 - 4\alpha(1-\alpha)\nu]}{2\mu_\alpha}. \tag{11}$$

Proof. We condition on $\mathcal{F}_t$, under which f_t and s_t are fixed. Then

$$\mathbb{E}[\langle f_t, u_t\rangle \mid \mathcal{F}_t] = \alpha\langle f_t, \bar{g}_t\rangle + (1-\alpha)\sum_i f_{t,i}s_{t,i}m_{t,i},$$

$$\sum_i f_{t,i}s_{t,i}m_{t,i} = (1-2\psi_t)Z_t.$$

The second identity remains true when $Z_t = 0$. Since $Z_t \geq 0$, $\psi_t \leq \psi < 1/2$, and $m_{t,i} \geq |\bar{g}_{t,i}|$,

$$(1-2\psi_t)Z_t \geq (1-2\psi)Z_t \geq (1-2\psi)\big[(1-\kappa_m)\|f_t\|^2 - \delta_m\big].$$

Combining this inequality with direction retention and taking total expectations, we obtain

$$\mathbb{E}\langle f_t, u_t\rangle \geq \mu_\alpha \mathbb{E}\|f_t\|^2 - \delta_\alpha. \tag{12}$$

Since $1-\kappa$ and $(1-2\psi)(1-\kappa_m)$ are positive, their convex combination μ_α is also positive. By smoothness and Eq. (10),

$$\mathbb{E}F(w_{t+1}) \leq \mathbb{E}F(w_t) - \eta\mu_\alpha\mathbb{E}\|f_t\|^2 + \eta\delta_\alpha + \frac{L\eta^2G^2}{2}[1-4\alpha(1-\alpha)\nu]. \tag{13}$$

The second-moment bound ensures that the required expectations are finite over a finite horizon. We sum over $t = 0, \ldots, T-1$, use $\mathbb{E}F(w_T) \geq F^\star$, and divide by $\eta\mu_\alpha T$ to obtain Eq. (11). □

If the additive errors are zero and the constants hold uniformly in T, setting $\eta = c/\sqrt{T}$ for fixed $c > 0$ gives an $O(T^{-1/2})$ bound on the average expected squared gradient norm. Nonzero additive errors leave a floor of δ_α/μ_α, and a fixed positive step size retains the second-moment term. Under the stated assumptions, these results give bounds on stationarity; they do not establish convergence of the iterates or convergence to a global optimum.

G.5 Interpretation

Choosing the blend. The blend controls both alignment and update size. For fixed constants, μ_α is affine in α. The descent coefficient is therefore largest at $\alpha = 0$ when $(1-2\psi)(1-\kappa_m) \geq 1-\kappa$, and at $\alpha = 1$ when the inequality is reversed. With equality, all blends give the same coefficient. In contrast, the damping factor $1-4\alpha(1-\alpha)\nu$ is smallest at $\alpha = 1/2$ when $\nu > 0$, favoring partial blends when the two sources often disagree. Finding the optimum of the full bound requires the constants, since the bound divides by μ_α and includes the bias floor. These constants may themselves depend on α through the trajectory. We therefore use this decomposition to explain the tradeoff in Table 6. It does not predict the best blend or justify using the blend from the predecessor's spectral correction.

Delay, EMA history, and transport. For a checkpoint from a fixed index $\tau \leq t$, the plain-step model gives

$$\|w_t - w_\tau\| \leq \eta\sum_{j=\tau}^{t-1}\|u_j\|, \qquad \mathbb{E}\|w_t - w_\tau\| \leq \eta(t-\tau)G.$$

The second inequality follows from Lemma 4 and Cauchy–Schwarz. This upper bound does not imply that the actual lag or ψ_t changes monotonically. Each EMA contribution has its own age in Lemma 5, so the latest checkpoint delay alone cannot describe the history. For reconstructed anchor weights, we separate delay and reconstruction error using $\|\hat{w}_j - w_t\| \leq \|w_{\tau_j} - w_t\| + \|\hat{w}_j - w_{\tau_j}\|$. Relating reconstruction error to an error-feedback buffer requires the actual refresh protocol.

Lossless sign-change coding preserves the correction if the receiver reconstructs the same ordered fields. Sparse checkpoint refresh may change these fields through reconstruction error, while basis staleness may change both fast-gradient retention and sign disagreement. Thus, delay affects more than ψ in the full analysis, and the theorem does not establish a smooth accuracy curve for the joint-staleness experiment. The analysis also excludes AdamW momentum, preconditioning, weight decay, and synchronization. The coordinatewise algebra holds for any tensor included in the modeled update, but applying a retention assumption to all model parameters is an additional requirement.

Remark 1 (RLVR and forecasting). The telescoping argument requires the same objective F at every step. With a fixed rollout batch, we can analyze a local surrogate if the sign measurability and alignment conditions hold. This does not allow us to telescope across the changing RLVR surrogates. An expected-return objective can itself be fixed, but our proof does not establish the required alignment with its gradient for the clipped GRPO updates, reused advantages, or forecast anchors used in our experiments. Evaluating the anchor on the current batch may also violate the conditioning used in the proof.

The deterministic discrepancy lemma still holds when we decompose each anchor gradient relative to the same reference objective. Differences between batches or surrogate objectives then enter ζ_j, so weight lag alone does not describe the discrepancy. Forecasting changes $\|\hat{w}_j - w_t\|$ and may reduce or increase it. A linear forecast is exact for an exactly linear trajectory with matched timing and gain, but we assume no forecast-error bound here. Reports of approximately linear RLVR trajectories (Wang et al., 2026; Wei et al., 2026) motivate this experiment; they do not establish improved sign fidelity or RLVR convergence for COMPASS.

H RLVR Configuration and Evaluation

Training and checkpoint protocol. We use GRPO for all reported RLVR comparisons, with 128 prompts per update, eight sampled responses per prompt and one PPO epoch. Table 14 lists the model and training settings. We train each configuration once, using the fixed hyperparameters in Section 6.4. The initial dense anchor makes no optimizer update. At the fixed cadence and delay, the first periodic anchor uses an exact admitted checkpoint at update 20, and forecasting begins at update 40. We run greedy validation every 100 updates, with no pre-training validation, and use the saved weights at each model's stated horizon for benchmark evaluation.

Both models use the same custom prompt template. GRPO uses 128 prompts and eight responses, a PPO minibatch of 128, a microbatch of one per GPU and one PPO epoch. Training sampling uses temperature 1 and top-p 1. AdamW uses learning rate 10^{-6}, a constant schedule, weight decay .01 and betas (.9,.999). We set PPO clipping to .2, gradient clipping to 1, the reference-KL loss coefficient to .001 and the entropy coefficient to 0. Validation uses greedy decoding every 100 updates, without pre-training validation. Since response budgets and GPU counts differ by model, cross-model differences cannot be attributed solely to the codec.

Table 14: RLVR model and training settings. Shared final hyperparameters are stated in Section 6.4.

Setting	Qwen2.5-Math-1.5B	Qwen3-4B-Base
Training dataset	MATH	MATH
Reported training horizon	200	100
Trainer GPUs per run	1	2
Rollout tensor parallelism	1	1
Maximum prompt tokens	1024	1024
Training / validation response tokens	2,048	3,072
Benchmark evaluation response tokens	3,072	3,072
Learned-Q rank (seven boundaries)	77	128
Validation examples after filtering	499	499

Selection and reporting scope. We use the sign-blend and basis-memory screening on 1.5B to choose the fixed learned-Q recipe. Validation observations also inform the reported horizons. We do not reselect the recipe using the subsequent component controls, codec comparisons or benchmark evaluations. The main table compares the transferred learned-Q configuration with Base and Dense at each scale, while the ablations below use only 1.5B. For 4B, the loader has 7491 eligible prompts and 58 full batches per epoch, so 100 updates correspond to approximately 1.72 epochs.

Benchmark protocol. We report greedy accuracy on MATH500, GSM8K, Minerva, OlympiadBench and MMLU-STEM. For AMC23, AIME24/25/26 and HMMT25, we sample eight responses per prompt with temperature .7 and top-p .8, and report the fraction of all responses that are correct (mean@8). We weight benchmarks equally when computing means. To check correctness, we extract the last boxed answer with the verifier and test exact equality after fixed MATH string normalization. Missing or unparseable boxed answers receive zero. Both model-specific benchmark suites use 3072 response tokens. MMLU-STEM contains 40 examples from each of 19 subjects, and OlympiadBench retains 672 of 674 prompts after filtering. MATH500 has 46 exact validation-prompt matches for

both models. We use complete source-specific suites and verify the saved responses for all benchmark results.

Codec controls and payload. We keep the optimizer and anchor/sign recipe fixed in the 1.5B codec ablations. PRF sends 77 retained BF16 coordinates selected by a shared keyed mask, while learned Q sends 77 BF16 coefficients. Both require 1232 nominal bits per token, boundary and direction. Our **PRF-subsampled TAH-Quant adaptation** (He et al., 2025) selects 256 of the 1536 coordinates and quantizes them as eight 32-coordinate tiles. We use tile entropy to choose which four tiles receive INT4 and which four receive INT3, and apply conditional Hadamard rotation with threshold 2 and nearest rounding. The nominal budget assigns 16 bits each to scale and offset, plus 7 bits to precision, pivot and rotation, giving 39 metadata bits per tile. The resulting payload is $4(32)(4) + 4(32)(3) + 8(39) = 1208$ bits, a nominal 95.085% reduction from $1536 \times 16 = 24576$ dense BF16 bits. The receiver regenerates coordinate indices from the shared PRF key, scatters the reconstructed coordinates and rescales by $1536/256$.

For the **learned-Q + TAH-Quant adaptation**, we quantize all 256 coefficients of an anchor-learned basis. Let $\mathcal{T}$ denote TAH quantization and reconstruction. The sender transmits $\mathcal{T}(hQ)$, and the receiver reconstructs $\mathcal{T}(hQ)Q^\top$. An incoming activation gradient g is reconstructed as $\mathcal{T}(gQ)Q^\top$. We use neither PRF subsampling nor inverse-keep rescaling. The same tile settings give the same nominal 1208-bit budget and 95.085% reduction. Compared with the main rank-77 BF16 method, this changes both retained rank and coefficient precision. The larger basis also increases basis-refresh traffic. In both adaptations, we compress only through the trainer interface; rollout, anchor and evaluation computation remain dense. These nominal boundary budgets exclude basis updates and other interfaces, and do not establish network speedup.

Metadata precision in the emulation. In both TAH adaptations, we compute and retain scale and offset in FP32 without rounding them to the 16-bit fields assumed above. Counting these fields as FP32 in the same layout gives $896 + 8(64 + 7) = 1464$ bits, or a 94.043% reduction. Although the 1208-bit budget is 1.95% below Q/PRF, we do not emulate its metadata rounding. The reported accuracy therefore does not validate a serialized 1208-bit implementation. Neither calculation represents measured network traffic.

I RLVR Ablations on Qwen2.5-Math-1.5B

We conduct all ablations in this section on Qwen2.5-Math-1.5B using MATH. We report validation scores at 100 and 200 updates and evaluate the codecs on the benchmarks at 200 updates. We select the learned-Q recipe through validation screening and keep it fixed for the subsequent controls.

Codec comparison. We compare all codecs on the complete benchmark suite in Table 15; Appendix H distinguishes nominal budgets from the metadata precision used in emulation. Learned Q performs better than PRF on the ten-task mean. Quantizing PRF-selected coordinates or learned-Q coefficients gives means of 30.08% and 30.12%, respectively, close to learned Q at 30.03%. The learned-Q + TAH variant reaches 67.60% on MATH500, but its nine-task OOD mean is 25.96%, compared with 26.01% for learned Q. We report validation scores in Table 11 and retain the fixed main configuration.

Table 15: Codec ablation on Qwen2.5-Math-1.5B. Benchmark results (%) at 200 updates with the shared anchor/sign recipe. TAH-Quant operates on PRF-selected coordinates or learned-Q coefficients. All columns use complete ten-task suites, dense inference and a 3,072-token response budget. Competition scores use mean@8 and other tasks use greedy accuracy, with benchmarks weighted equally in the means. Nominal budgets are 1,232 bits for Q/PRF and 1,208 for both TAH adaptations per token, boundary and direction. This TAH budget assumes 16-bit scale/offset metadata, while emulation retains FP32 metadata (1,464 bits if counted as FP32). Appendix H gives the accounting, evaluation protocol and MATH500 validation overlap.

Benchmark	Learned Q ↑	PRF ↑	PRF + TAH-Quant ↑	Learned Q + TAH-Quant ↑
MATH500 (ID)	66.20	63.60	64.80	**67.60**
GSM8K	77.48	76.19	**80.06**	79.15
Minerva	15.81	15.44	**16.54**	15.81
OlympiadBench	21.73	20.54	22.32	**22.62**
MMLU-STEM subset	**45.66**	32.76	40.53	41.97
AMC23	44.69	**52.81**	50.31	50.31
AIME24	12.92	8.75	**13.33**	12.50
AIME25	**7.08**	5.42	5.83	5.83
AIME26	7.92	**10.83**	7.08	5.42
HMMT25	**0.83**	**0.83**	0.00	0.00
Ten-benchmark mean	30.03	28.72	30.08	**30.12**
Nine-OOD mean	26.01	24.84	**26.22**	25.96

Sign blend, basis memory and reuse. In Table 16, we vary α and β_{anc} while keeping $\beta_Q = .9$ and using each sign field once. We then vary β_Q and sign reuse in Table 12 in the main experiments. Repeated reuse applies the latest sign field at every update, whereas single use consumes a fresh field once. Annealed reuse weights the correction by $.75^a$ at age a. We use these validation comparisons to choose the fixed repeated-sign learned-Q configuration.

Table 16: Validation accuracy (%) for sign-blend and anchor-memory screening. Columns identify the evaluation update. All settings use covariance-EMA Q with $\beta_Q = .9$ and single-use signs. We compare validation readings at the shared 100- and 200-update checkpoints, using these screening runs to inform the subsequent fixed configuration.

α	β_{anc}	Val@100 ↑	Val@200 ↑
0.25	0.25	**63.93**	56.11
0	0	63.73	62.73
0	0.25	63.73	**62.93**
0	0.5	62.32	58.92
0.25	0	61.32	55.51
0.25	0.5	63.13	58.92
0.5	0	58.72	58.32
0.5	0.25	63.33	59.12
0.5	0.5	63.73	57.72

Anchor guidance and weight forecasting. We compare anchor/sign controls in Table 11(b). In Table 17, we test forecast gain at matched delays and compare basis estimators. We forecast anchor weights by secant extrapolation (Eq. (5)), which is distinct from the activation projection. At $K = 40$, gain 1 gives 66.33% validation accuracy, compared with 63.33% for gain .5. At $K = 80$, gains 1 and .25 give 65.33% and 65.53%. These comparisons measure sensitivity to forecast strength at fixed delay. The frozen-basis comparison changes anchor guidance and signs jointly.

(a) We keep cadence at 20 and vary gain at matched delays of 40 and 80, using Eq. (5) for forecasting. The first forecasts occur at updates 20/40/60/100 for delays 10/20/40/80; earlier fires use exact snapshots. *(b)* Covariance EMA uses two subspace iterations, while the fresh-anchor sketch uses one and has no β_Q. This comparison therefore does not isolate β_Q, whereas the two sketch rows isolate sign reuse.

Table 17: Qwen2.5-Math-1.5B learned-Q component controls at 200 updates, showing validation accuracy (%) on the same 499 examples. Val@100 and Val@200 identify the evaluation update. Bold compares forecast gains at the same delay and sign reuse with the same basis estimator. The shared fixed reference uses an adaptive covariance-EMA basis ($\beta_Q = .9$), repeated signs, delay $K = 20$ and forecast gain 1. Panels vary (a) delay and forecast gain and (b) the basis estimator and sign reuse. Table 11(b) gives the anchor, basis and sign controls.

Configuration	Setting	Val@100 ↑	Val@200 ↑
Fixed reference	Adaptive basis; repeated signs	62.93	63.33
(a) Delay and forecast-gain controls			
$K = 10$	$\gamma = 1$	65.93	66.93
$K = 40$	$\gamma = 1$	**66.33**	**66.33**
$K = 80$	$\gamma = 1$	63.93	65.33
$K = 40$	$\gamma = 0.5$	65.33	63.33
$K = 80$	$\gamma = 0.25$	**64.93**	**65.53**
(b) Basis-estimator and sign-reuse controls			
Fresh-anchor sketch	Repeated signs	–	**66.53**
Fresh-anchor sketch	Signs once	–	50.30