TEMPER: READING THE SECOND MOMENT FROM THE RIEMANNIAN GRADIENT IN LORA

Anonymous authors
Paper under double-blind review

ABSTRACT

Low-rank adaptation trains a weight update as the product of two factors, and one update has many factorizations. Recent LoRA optimizers are therefore built to be transformation-invariant: the step should not depend on which one is stored. One line secures that by stepping along the Riemannian gradient, the projection of the Euclidean gradient onto the manifold's tangent space and a function of the update alone. However, invariance fixes the direction of the step, not how far it moves along each coordinate. That distance is what a second moment sets, and the methods that step along the Riemannian gradient either carry none or read one from the Euclidean gradient, almost all of which lies where no rank-r step can follow. To this end, we propose **Temper**, which accumulates one second moment per row and one per column of the Riemannian gradient itself, and never forms the dense $m \times n$ matrix that reading it there is known to cost: a tangent vector is low rank, so both marginals follow exactly from two small Gram matrices. Temper is provably invariant and costs no more per step than an unpreconditioned one. On Llama-3-8B and Qwen-2.5-7B it leads the strongest baseline by 0.9 to 3.2 points on the commonsense average, GSM8K and all three HumanEval pass rates.

1 INTRODUCTION

Low-Rank Adaptation (LoRA) (Hu et al., 2022) is the standard way to fine-tune a large language model under a parameter budget. For a frozen weight $W \in \mathbb{R}^{m\times n}$ it learns an additive update $Z = BA$ with $B \in \mathbb{R}^{m\times r}$, $A \in \mathbb{R}^{r\times n}$; we call A and B the LoRA factors. Only the product Z changes the network's function, but the optimizer sees the two factors A and B. The same Z has infinitely many factorizations, so an optimizer that reads the factors can take two different steps from the same starting function. An optimizer whose step is the same for every factorization is transformation-invariant (Yen et al., 2025); Adam (Kingma and Ba, 2015) is not.

Yen et al. (2025) prove that no diagonal preconditioner on the factor gradients can be invariant, and secure invariance with an $r \times r$ matrix preconditioner on the unmagnified gradients, reporting gains across tasks and model sizes from securing it. The second construction leaves the factors behind: the rank-r matrices form a smooth manifold, so Z can be optimized on it directly, along the Riemannian gradient $G = P_Z(\nabla Z)$, the orthogonal projection of ∇Z onto the tangent space at Z (Vandereycken, 2013). LORO (Mo et al., 2025), Riemannion (Bogachev et al., 2026), LoRA-Muon (Cesista et al., 2026) and RAdamW (Bian et al., 2025) all step in such a direction, where every quantity is a function of Z alone and invariance holds whichever pair stores Z.

However, invariance settles which direction to step in, not how far to move along each coordinate of it. That is what an adaptive second moment answers, and on the fixed-rank manifold it is answered badly. Of the methods that step along G, Riemannion and LoRA-Muon carry no adaptive scales at all and LORO takes its scales from the factor gradients; RAdamW carries one per row and one per column, but reads them from ∇Z through a backward hook on the merged weight. A scale read there does not describe the direction the step takes: the normal part $N = \nabla Z - G$ is invisible to every rank-r step, and it is most of the gradient. Across the 28 adapted modules of a Llama-3.2-3B run at $r = 16$, between 0.93 and 0.99 of $\|\nabla Z\|_F^2$ sits in N. The little that remains is not a usable stand-in either: across those modules a tenfold change in $\rho(\nabla Z)_i$ moves $\rho(G)_i$ by only about a factor of two, so a scale read from the first is calibrated to a spread the second does not have (Figure 1). A second moment is only meaningful for the matrix it scales, and on this manifold that matrix is G.

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.

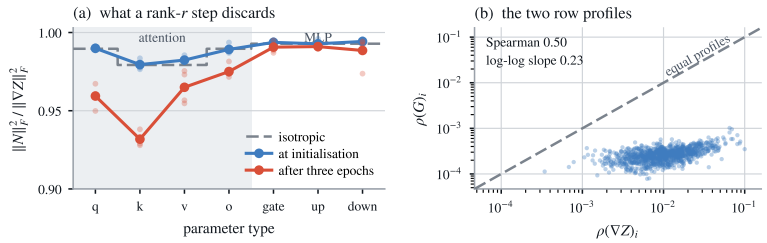

Figure 1: (a): the share of $\|\nabla Z\|_F^2$ carried by N, the component no rank-r step can use with the isotropic reference (5). (b): the two row profiles on one layer, one point per row.

To this end, we propose Temper, a LoRA optimizer that gives every row and every column of the update its own second moment, read from the Riemannian gradient it steps along (Figure 2). Row i of G is output dimension i of the layer and column j is input dimension j at every step, and neither index depends on the factorization, so an elementwise statistic read from G is invariant (Proposition 1, Theorem 1); the impossibility result of Yen et al. (2025) ranges over diagonal preconditioners of ∇A and ∇B, and does not reach it. Reading a second moment there is known to cost the Riemannian gradient itself: Kasai et al. (2019) and Bécigneul and Ganea (2019) accumulate the same statistic where the manifold point is already thin and forming it is free, but here it is an $m \times n$ matrix. Temper does not pay that price. A tangent vector has rank at most $2r$ and factors as $G = PQ^\top$ with P and Q thin, so both sums follow exactly from two $2r \times 2r$ Gram matrices, at $O((m+n)r^2)$ per step, from the factor gradients autograd already returns and with no $m \times n$ matrix formed.

Our contributions are:

- We show that transformation invariance fixes the direction of a LoRA step but not the distances moved along each of its coordinates, and that the second moment which sets those distances has to be read from the direction actually taken rather than from the Euclidean gradient.
- We propose Temper, which accumulates a row and column second moment on the Riemannian gradient it steps along, and prove its update transformation-invariant. Both sums follow exactly from two $2r \times 2r$ Gram matrices, so no $m \times n$ matrix is formed.
- We evaluate Temper on Llama-3-8B and Qwen-2.5-7B against LoRA with AdamW, LoRA-RITE, LORO, RAdamW and Riemannion. It is first on 23 of the 24 task–model cells, gaining 0.9 to 3.2 points on the commonsense average, GSM8K and HumanEval.

2 Two Requirements on the LoRA Update

2.1 Why the Update Must Be Transformation-Invariant

Let f be the fine-tuning loss. Write $\nabla Z := \partial f/\partial Z$, $\nabla A := \partial f/\partial A$ and $\nabla B := \partial f/\partial B$, so that

$$\nabla B = \nabla Z\, A^\top, \qquad \nabla A = B^\top \nabla Z. \tag{1}$$

The model sees only the product $Z = BA$, so for any invertible $R \in \mathbb{R}^{r\times r}$ the pair $(BR,\ R^{-1}A)$ stores the same Z and the loss cannot tell the two apart. The factor gradients can: by (1) the change replaces ∇B by $\nabla B\, R^{-\top}$ and ∇A by $R^\top \nabla A$. An optimizer that reads them without correcting for R moves Z differently depending on how Z is stored, which in LoRA is decided by the initialization and the constant α/r, neither of which knows the task.

Definition 1 (transformation invariance, Yen et al., 2025)**.** Let (B_1, A_1) be LoRA factors and let $B_2 = B_1 R$, $A_2 = R^{-1}A_1$ for an invertible $R \in \mathbb{R}^{r\times r}$, so that $B_1A_1 = B_2A_2$. An optimizer is transformation-invariant if its updates satisfy $(B_1 + \delta B_1)(A_1 + \delta A_1) = (B_2 + \delta B_2)(A_2 + \delta A_2)$.

Yen et al. (2025) prove that no preconditioner of the form $\mathrm{vec}(\delta A) = -\eta\, X\, \mathrm{vec}(\nabla A)$ with X diagonal can satisfy Definition 1; Adam on the factors is such a preconditioner. Two constructions do. The first stays in factor coordinates, resting on the polar decompositions $B = U_B R_B$ and $A^\top = U_A R_A$, with $U_B \in \mathbb{R}^{m\times r}$, $U_A \in \mathbb{R}^{n\times r}$ orthonormal and R_A, R_B symmetric positive semidefinite, and on the *unmagnified gradients*

$$\bar\nabla B := \nabla Z\, U_A = \nabla B\, R_A^\dagger, \qquad \bar\nabla A := U_B^\top \nabla Z = R_B^\dagger\, \nabla A, \tag{2}$$

Table 1: Optimizers by the second moment they accumulate, the object it is read from, and the number of adaptive scales it yields per adapted layer. "Time per step" is the work beyond the two factor gradients.

		Second moment			
Method	**Invariant**	**Form**	**Read from**	**# scales**	**Time per step**
Adam (Kingma and Ba, 2015)	no	elementwise	$\nabla B,\ \nabla A$	$(m{+}n)r$	$O((m{+}n)r)$
LoRA-Pro (Wang et al., 2025)	no	elementwise	$\tilde{g}$	mn	$O(mnr)$
LORO (Mo et al., 2025)	no	none	—	0	$O((m{+}n)r^2)$
LoRA-RITE (Yen et al., 2025)	yes	$r \times r$ matrix	$\bar{\nabla} B,\ \bar{\nabla} A$	$r(r{+}1)$	$O((m{+}n)r^2)$
RAdamW (Bian et al., 2025)	yes	row & column	∇Z	$m{+}n$	$O(mnr)$
Riemannion (Bogachev et al., 2026)	yes	none	—	0	$O((m{+}n)r^2)$
Temper (ours)	yes	row & column	G	$m{+}n$	$O((m{+}n)r^2)$

which depend on $\mathrm{col}(B)$ and $\mathrm{col}(A^\top)$ but not on the factor magnitudes; LoRA-RITE preconditions them with an $r \times r$ matrix. The columns of U_A and U_B are the *frame*, and a statistic read from $\bar{\nabla} B$ or $\bar{\nabla} A$ is in *frame coordinates*.

The second leaves factor coordinates altogether. Let $\mathcal{M}_r$ be the manifold of rank-r matrices and P_Z the orthogonal projection onto its tangent space at Z,

$$P_Z(Y) = U_B U_B^\top Y + (I - U_B U_B^\top)\, Y\, U_A U_A^\top. \tag{3}$$

We call $G := P_Z(\nabla Z)$ the *Riemannian gradient* of f at Z. Since P_Z depends on the factors only through $U_B U_B^\top$ and $U_A U_A^\top$, which Definition 1 leaves unchanged, G is a function of Z and ∇Z alone (Proposition 1), and a step along it is invariant whichever pair stores Z.

2.2 Why the update needs adaptive scales

The layer computes $h = (W + Z)\, x$, so $h_i = (Wx)_i + \sum_j Z_{ij} x_j$: row i of Z belongs to output i of the layer and column j to input j, and both indices are the layer's own. The rank index k in $Z_{ij} = \sum_k B_{ik} A_{kj}$ is not, since $(BR, R^{-1}A)$ leaves every Z_{ij} unchanged but mixes the k coordinates. A requirement on the update can therefore be stated per row and per column of Z, and not per k. A step along a direction D with one learning rate, $\delta Z = -\eta D$, moves row i by $\eta \|D_{i,:}\|_2$, in proportion to its gradient magnitude; we measure how unevenly it is spread with the quantity of Yen et al. (2025), applied to the rows of Z:

$$\mu_i(\delta Z) = \frac{\|\delta Z_{i,:}\|_2}{\|Z_{i,:}\|_2}, \qquad \mathrm{spread}(\delta Z) = \frac{\max_i \mu_i(\delta Z)}{\min_i \mu_i(\delta Z)}, \tag{4}$$

so that a balanced update has $\mathrm{spread}(\delta Z) = 1$.

On a loss that separates over rows with curvatures $0 < w_{\min} \le w_i \le w_{\max}$ and $\kappa := w_{\max}/w_{\min}$, one learning rate reaches relative error ϵ on every row in $\Theta\big(\kappa \log(1/\epsilon)\big)$ steps, while one scale per row, $\delta Z_{i,:} = -\eta\, d_i\, \nabla Z_{i,:}$ with $d_i = 1/w_i$, removes the dependence on κ (Appendix A). Equation (4) reads that ratio off the update rather than off the loss, and adaptive optimizers estimate the d_i from running averages of squared gradients (Kingma and Ba, 2015; Shazeer and Stern, 2018).

Three properties of the scales decide whether this mechanism survives on a low-rank adapter; Table 1 records them per optimizer.

Invariance. The scales are functions of the gradient they are read from, and by Section 2.1 the factor gradients change with the stored pair, as does their product $\nabla B\, A + B\, \nabla A = \nabla Z\, A^\top A + B B^\top \nabla Z$, the substitute for ∇Z that needs no hook on the merged weight. Scales read from either are not invariant, and neither is the balance they produce.

What they are read from. A scale is meaningful only for the direction it multiplies, and on $\mathcal{M}_r$ the step is a tangent vector while the Euclidean gradient is not. Writing $\nabla Z = G + N$ with $N := \nabla Z - P_Z(\nabla Z)$, the normal part satisfies $N A^\top = 0$ and $B^\top N = 0$ (Proposition 3), so no

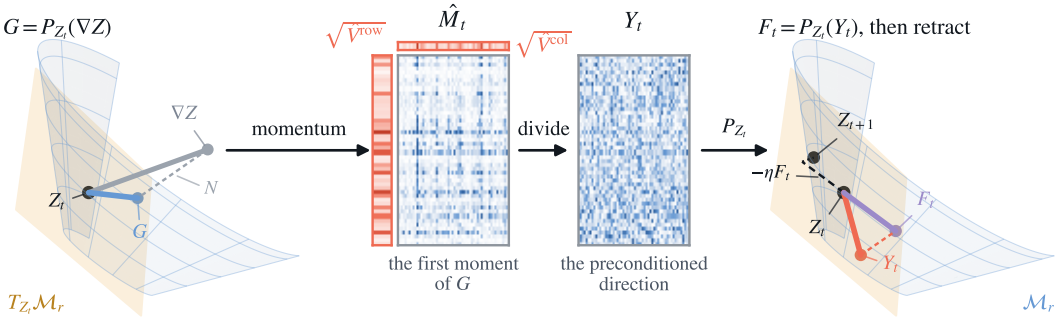

Figure 2: One step of Temper: the Riemannian gradient $G = P_{Z_t}(\nabla Z)$ feeds a first moment, which is divided by $\sqrt{\hat{V}_t^{\text{row}}}$ along the rows, by $\sqrt{\hat{V}_t^{\text{col}}}$ along the columns, projected back and retracted.

rank-r step can use it, and it is not a small remainder: for an isotropic ∇Z,

$$\frac{\mathbb{E}\|N\|_F^2}{\mathbb{E}\|\nabla Z\|_F^2} = \frac{(m-r)(n-r)}{mn}, \tag{5}$$

close to one whenever $r \ll \min(m,n)$, and Figure 1(a) confirms it on Llama-3.2-3B at $r = 16$: every parameter type starts on the reference and stays above 0.93 after three epochs, the attention projections moving most. Magnitudes read from ∇Z therefore describe mainly what the step throws away, and they do not order the rows the way $\rho(G)$ does (Figure 1(b)).

Number of scales. The row and column magnitudes of a rank-r update carry $m + n$ degrees of freedom. A preconditioner on the rank index,

$$\delta B = -\eta\, \bar{\nabla} B\, C_B, \qquad \delta A = -\eta\, C_A\, \bar{\nabla} A, \qquad C_A, C_B \in \mathbb{R}^{r\times r}, \tag{6}$$

as in Yen et al. (2025), multiplies every row of $\bar{\nabla}B$ by the same C_B and every column of $\bar{\nabla}A$ by the same C_A. Row i of $\bar{\nabla}B$ is output dimension i, since $Z_{i,:} = B_{i,:}A$, so two rows that differ in gradient magnitude stay separated by a comparable factor after the step; and the symmetric pair carries $r(r+1)$ free numbers, which for $r \ll \min(m,n)$ cannot set $m+n$ magnitudes independently.

3 Temper

Temper steps along G, and never builds it. Substituting (2) into (3) gives the standard factored form,

$$G = \dot{B}\, U_A^\top + U_B\, \dot{A} = PQ^\top, \qquad P = [\,\dot{B},\ U_B\,] \in \mathbb{R}^{m\times 2r}, \qquad Q = [\,U_A,\ \dot{A}^\top\,] \in \mathbb{R}^{n\times 2r}, \tag{7}$$

with $\dot{B} = (I - U_B U_B^\top)\,\bar{\nabla}B$ and $\dot{A} = \bar{\nabla}A$. A tangent vector has rank at most $2r$, so the pair $(\dot{B}, \dot{A})$ is G: it is assembled from the two unmagnified gradients of (2), which the factor gradients already give, and the $m \times n$ matrix is never allocated. Figure 2 visualizes the workflow of our method.

3.1 Why the statistic comes from G

Under Definition 1 the column spaces of B and $A^\top$ do not change, so the frames can only rotate: $U_{A_2} = U_{A_1} Q$ for an orthogonal Q. The unmagnified gradient inherits the rotation,

$$\bar{\nabla}B_2 = \nabla Z\, U_{A_2} = \nabla Z\, U_{A_1} Q = \bar{\nabla}B_1\, Q. \tag{8}$$

A matrix second moment is unaffected by Q: the two rotations in $\bar{\nabla}B_2^\top \bar{\nabla}B_2 = Q^\top(\bar{\nabla}B_1^\top \bar{\nabla}B_1)Q$ cancel when the preconditioned step $\bar{\nabla}B(\bar{\nabla}B^\top\bar{\nabla}B)^{-1/2}$ is formed. This is why LoRA-RITE works. An elementwise second moment is not so fortunate, because squaring entry by entry does not commute with multiplication by Q:

$$\bar{\nabla}B_2 \odot \bar{\nabla}B_2 = (\bar{\nabla}B_1 Q) \odot (\bar{\nabla}B_1 Q) \ \neq\ (\bar{\nabla}B_1 \odot \bar{\nabla}B_1)\, Q. \tag{9}$$

The same holds for $\bar{\nabla}A$ and for the raw factor gradients. The obstruction is Q, which appears only because the frames are bases the factorization may rotate; G carries none.

Proposition 1. *G does not change under the transformation of Definition 1. Every function of G read as an $m \times n$ matrix is therefore transformation-invariant, elementwise functions included.*

The proof is two lines and is in Appendix A: P_Z enters only through the projectors $U_A U_A^\top$ and $U_B U_B^\top$ onto the row and column spaces of Z, which Definition 1 fixes. An elementwise second moment is therefore permitted, provided it is read from G rather than from the factor gradients.

3.2 THE PRECONDITIONER

One entry per coordinate of G would cost mn numbers per layer, which defeats the purpose of a low-rank adapter. Following Adafactor (Shazeer and Stern, 2018), we keep the row and column sums of the squared entries,

$$\rho(G)_i = \sum_j G_{ij}^2, \qquad \gamma(G)_j = \sum_i G_{ij}^2, \tag{10}$$

and accumulate each as an exponential moving average,

$$V_t^{\mathrm{row}} = \beta_2 V_{t-1}^{\mathrm{row}} + (1-\beta_2)\,\rho(G_t), \qquad V_t^{\mathrm{col}} = \beta_2 V_{t-1}^{\mathrm{col}} + (1-\beta_2)\,\gamma(G_t), \tag{11}$$

so the state is $m+n$ numbers per layer. Neither $\rho(G_t)$ nor $\gamma(G_t)$ needs G_t. The row sums of $G \odot G$ are the diagonal of $GG^\top = P(Q^\top Q)P^\top$ and the column sums the diagonal of $G^\top G = Q(P^\top P)Q^\top$, so

$$\rho(G) = \mathrm{diag}\left(P\,(Q^\top Q)\,P^\top\right), \qquad \gamma(G) = \mathrm{diag}\left(Q\,(P^\top P)\,Q^\top\right), \tag{12}$$

as identities, not approximations. Both diagonals are formed as the row sums of $(PS)\odot P$ with $S = Q^\top Q$ and of $(QT)\odot Q$ with $T = P^\top P$, so neither $PSP^\top$ nor G is ever allocated: $O((m+n)r^2)$ against the $O(mnr)$ and the transient mn numbers that forming G would cost.

We write $\hat{V} = V/(1-\beta_2^t)$ for the bias correction and add a small ϵ to every entry of $\hat{V}_t^{\mathrm{row}}$ and $\hat{V}_t^{\mathrm{col}}$, rather than to the reconstructed matrix, which would destroy the rank-one structure we rely on below. Take the update without momentum first. Adafactor's rank-one reconstruction combines the two vectors into

$$\widehat{V}_{t,ij} = \frac{\hat{V}_{t,i}^{\mathrm{row}}\;\hat{V}_{t,j}^{\mathrm{col}}}{\sum_i \hat{V}_{t,i}^{\mathrm{row}}}, \tag{13}$$

and the preconditioned step is $\delta Z = -\eta\, P_{Z_t}\big(G_t/\sqrt{\widehat{V}_t}\big)$. Since $\widehat{V}_t$ is a rank-one outer product, dividing by its square root entry by entry is a diagonal scaling on each side:

$$\frac{Y}{\sqrt{\widehat{V}_t}} = c_t\, D_t^{\mathrm{row}}\, Y\, D_t^{\mathrm{col}}, \qquad (D_t^{\mathrm{row}})_{ii} = (\hat{V}_{t,i}^{\mathrm{row}})^{-1/2}, \quad (D_t^{\mathrm{col}})_{jj} = (\hat{V}_{t,j}^{\mathrm{col}})^{-1/2}, \tag{14}$$

with $c_t = (\sum_i \hat{V}_{t,i}^{\mathrm{row}})^{1/2}$ a single scalar restoring the overall scale. Index i is output dimension i of the layer and index j is input dimension j, at every step; a coordinate of the frame carries no such fixed meaning, because the frame is recomputed at every step and may be rotated between one step and the next. Proposition 2 states what the two divisions do to the row profile, for a single step, so that $\hat{V}^{\mathrm{row}} = \rho(G)$ and $\hat{V}^{\mathrm{col}} = \gamma(G)$; the scalar c_t is common to all rows and does not enter a ratio.

Proposition 2. *Let $\rho_i = \rho(G)_i > 0$ and $\gamma_j = \gamma(G)_j > 0$, and let $Y = D^{\mathrm{row}} G\, D^{\mathrm{col}}$ with $D^{\mathrm{row}} = \mathrm{diag}(\rho^{-1/2})$ and $D^{\mathrm{col}} = \mathrm{diag}(\gamma^{-1/2})$. Then every row of Y satisfies*

$$\frac{1}{\max_j \gamma_j} \;\le\; \|Y_{i,:}\|_2^2 \;\le\; \frac{1}{\min_j \gamma_j}, \qquad \textit{hence} \qquad \frac{\max_i \|Y_{i,:}\|_2}{\min_i \|Y_{i,:}\|_2} \;\le\; \left(\frac{\max_j \gamma_j}{\min_j \gamma_j}\right)^{1/2}. \tag{15}$$

Neither bound involves ρ.

Proof. With $w_{ij} = G_{ij}^2/\rho_i$ we have $\|Y_{i,:}\|_2^2 = \sum_j w_{ij}\,\gamma_j^{-1}$, where $w_{ij} \ge 0$ and $\sum_j w_{ij} = 1$ by the definition of ρ_i, and a convex combination of $\{\gamma_j^{-1}\}$ lies between the smallest and the largest of them. □

The content is the absence of ρ on the right: dividing row i by $\rho_i^{1/2}$ leaves a row of unit length, so the row profile of G is removed and what remains is the column profile, a different quantity. Contrast (6), where the same linear map is applied to every row and the bound is the input row profile itself.

The proof also says where the two vectors may be read from, since $\sum_j w_{ij} = 1$ holds only because ρ_i is a statistic of the matrix the scales multiply. Reading ρ and γ from ∇Z while the matrix scaled is still G multiplies row i by $\kappa_i := \rho(G)_i/\rho(\nabla Z)_i$ and loosens (15) by $(\kappa_{\max}/\kappa_{\min})^{1/2}$, re-admitting the row profile that Proposition 2 removes (Appendix A).

Two steps remain between Proposition 2 and the update. The preconditioned gradient is projected back onto the tangent space, which mixes rows: it moves row i by at most $\Pi_{ii}^{1/2}\sigma_{\max}(Y)$, and the leverage scores $\Pi_{ii} = (U_B U_B^\top)_{ii}$ sum to r over the m rows, so only a few rows move appreciably (Proposition 5, Appendix A). Second, (14) maps $P \mapsto D_t^{\mathrm{row}} P$ and $Q \mapsto D_t^{\mathrm{col}} Q$, so the preconditioned step still has rank at most $2r$ and retracts at the cost of an unpreconditioned one.

3.3 The full step

G_t is a tangent vector at Z_t while the stored moment M_{t-1} is one at Z_{t-1}, so adding them directly is incorrect. We keep the first moment and pay for transporting it, using projection as the transport (Vandereycken, 2013):

$$M_t = \beta_1 P_{Z_t}(M_{t-1}) + (1-\beta_1)G_t, \qquad \hat{M}_t = M_t/(1-\beta_1^t). \tag{16}$$

Our second moment is indexed by the rows and columns of the layer, which do not move, so it needs no transport. The transport in (16) runs on the stored pair $(\dot{M}_B, \dot{M}_A)$, where $M_{t-1} = \dot{M}_B \tilde{U}_A^\top + \tilde{U}_B \dot{M}_A$ with $\tilde{U}_B^\top \dot{M}_B = 0$. Writing P_{Z_t} in that form takes four $r \times r$ products, the frame overlaps $\tilde{U}_A^\top U_A$ and $U_B^\top \tilde{U}_B$ together with $U_B^\top \dot{M}_B$ and $\dot{M}_A U_A$, and returns a pair of the same shape at $O((m+n)r^2)$ with no $m \times n$ matrix formed (Appendix A).

The update rule, with both moments, is

$$F_t = P_{Z_t}\left(\frac{\hat{M}_t}{\sqrt{\hat{V}_t}}\right), \qquad Z_{t+1} = \mathrm{SVD}_r\left(Z_t - \eta\, F_t\right). \tag{17}$$

Both Z_t and F_t have the form (7), so SVD_r runs on a $2r \times 2r$ core. The second moment is read from G_t while the matrix it divides in (17) is $\hat{M}_t$, which is Adam's own ordering. Proposition 2 is a single-step statement, where the two coincide; with momentum the ratio $\rho(\hat{M}_t)_i/\rho(G_t)_i$ re-enters (15), bounded by how far the row profile of G moves over the $1/(1-\beta_1)$ steps the average spans. That is a different question from which matrix the statistic is read from: reading it from ∇Z is already wrong at $\beta_1 = 0$.

Theorem 1. *Assume the LoRA factors keep rank r along the trajectory, and work in exact arithmetic. Then Temper is transformation-invariant in the sense of Definition 1. For any invertible R applied to the initial factors, the quantities G_t, V_t^{row}, V_t^{col}, $\hat{M}_t$ and F_t agree at every step, and so does $\{Z_t\}$.*

The proof is an induction on t: Proposition 1 makes G_t agree, every later quantity is a function of G_t read as a matrix or of Z_t, and SVD_r depends on $Z_t - \eta F_t$ alone. Appendix B gives it, including the effect of a change of frame on the factors P and Q of (7).

The same induction gives Corollary 1 in Appendix B: the convention used to split Z_{t+1} back into factors does not change $\{Z_t\}$, so writing $B \leftarrow U\Sigma^{1/2}$, $A \leftarrow \Sigma^{1/2}V^\top$ and writing $B \leftarrow U\Sigma$, $A \leftarrow V^\top$ give the same trajectory. Theorem 1 does not contradict Yen et al. (2025): their result rules out diagonal preconditioners applied to ∇A and ∇B, while our statistic is elementwise on G, and the step from the factor gradients to G passes through $R_A^\dagger$ and $R_B^\dagger$, which are not diagonal. The full-rank assumption is where finite precision enters: R_A and R_B are inverted through a pseudo-inverse with a small cutoff, and invariance is exact as long as that cutoff stays inactive. Appendix C collects the step as Algorithm 1 and refer to Appendix D for complexity analysis.

Table 2: Commonsense reasoning accuracy (%) on eight benchmarks, mean and standard deviation over three seeds. "Avg." is the unweighted mean of the eight task means. Best results per model are in **bold** and second best are underlined.

Method	# Params	BoolQ	PIQA	SIQA	HellaS	WinoG	ARC-e	ARC-c	OBQA	Avg.
LLaMA-3-8B										
LoRA	0.52%	$73.5_{\pm 1.4}$	$87.6_{\pm 0.4}$	$80.7_{\pm 0.4}$	$92.7_{\pm 0.4}$	$86.8_{\pm 0.4}$	$91.4_{\pm 0.4}$	$80.6_{\pm 0.4}$	$86.5_{\pm 0.4}$	85.0
LoRA-RITE	0.52%	$74.6_{\pm 1.1}$	$88.2_{\pm 0.7}$	$81.5_{\pm 1.2}$	$93.3_{\pm 1.0}$	$86.7_{\pm 0.8}$	$92.0_{\pm 0.6}$	$80.7_{\pm 1.4}$	$88.6_{\pm 1.2}$	85.7
LORO	0.52%	$73.8_{\pm 0.8}$	$87.5_{\pm 1.7}$	$80.4_{\pm 0.8}$	$91.6_{\pm 1.1}$	$85.7_{\pm 2.1}$	$91.1_{\pm 1.5}$	$78.4_{\pm 1.2}$	$85.8_{\pm 1.4}$	84.3
RAdamW	0.52%	$74.5_{\pm 0.6}$	$88.9_{\pm 1.0}$	$81.2_{\pm 1.3}$	$92.5_{\pm 1.6}$	$85.7_{\pm 1.1}$	$91.3_{\pm 1.5}$	$80.9_{\pm 1.1}$	$88.8_{\pm 1.2}$	85.5
Riemannion	0.52%	$71.4_{\pm 2.1}$	$86.3_{\pm 1.3}$	$79.4_{\pm 0.6}$	$88.6_{\pm 1.4}$	$81.3_{\pm 1.8}$	$90.3_{\pm 1.2}$	$79.9_{\pm 0.7}$	$84.3_{\pm 1.6}$	82.7
Temper	0.52%	$\mathbf{75.6}_{\pm 1.3}$	$\mathbf{89.7}_{\pm 1.2}$	$\mathbf{82.5}_{\pm 1.0}$	$\mathbf{94.8}_{\pm 2.3}$	$\mathbf{87.9}_{\pm 1.4}$	$\mathbf{93.2}_{\pm 0.8}$	$\mathbf{82.1}_{\pm 0.6}$	$\mathbf{89.6}_{\pm 2.0}$	**86.9**
Qwen-2.5-7B										
LoRA	0.53%	$74.4_{\pm 1.0}$	$89.4_{\pm 1.5}$	$81.8_{\pm 0.6}$	$93.8_{\pm 1.2}$	$87.2_{\pm 1.3}$	$93.3_{\pm 1.6}$	$86.6_{\pm 1.2}$	$90.0_{\pm 1.0}$	87.1
LoRA-RITE	0.53%	$\mathbf{75.0}_{\pm 2.0}$	$89.8_{\pm 1.0}$	$81.8_{\pm 1.7}$	$94.1_{\pm 0.7}$	$88.3_{\pm 1.2}$	$94.6_{\pm 1.5}$	$88.7_{\pm 0.6}$	$90.8_{\pm 0.5}$	87.9
LORO	0.53%	$72.1_{\pm 1.7}$	$87.4_{\pm 1.2}$	$80.5_{\pm 0.6}$	$90.9_{\pm 0.3}$	$85.7_{\pm 1.1}$	$92.1_{\pm 0.8}$	$85.8_{\pm 1.2}$	$89.7_{\pm 1.2}$	85.5
RAdamW	0.53%	$74.8_{\pm 1.0}$	$89.2_{\pm 0.7}$	$81.2_{\pm 1.1}$	$93.6_{\pm 0.3}$	$87.8_{\pm 2.0}$	$94.6_{\pm 1.5}$	$87.3_{\pm 0.9}$	$90.6_{\pm 1.0}$	87.4
Riemannion	0.53%	$70.5_{\pm 1.1}$	$88.6_{\pm 1.3}$	$81.0_{\pm 0.6}$	$92.7_{\pm 0.7}$	$86.6_{\pm 1.0}$	$92.7_{\pm 2.2}$	$87.5_{\pm 1.2}$	$90.1_{\pm 1.3}$	86.2
Temper	0.53%	$74.8_{\pm 1.6}$	$\mathbf{90.5}_{\pm 1.1}$	$\mathbf{82.7}_{\pm 1.0}$	$\mathbf{94.6}_{\pm 1.4}$	$\mathbf{89.5}_{\pm 0.7}$	$\mathbf{96.1}_{\pm 1.2}$	$\mathbf{89.7}_{\pm 1.6}$	$\mathbf{92.4}_{\pm 0.7}$	**88.8**

Table 3: Math reasoning (GSM8K, accuracy %) and code generation (HumanEval, Pass@k %), mean and standard deviation over three seeds.

		GSM8K	HumanEval		
Method	**# Params**	Accuracy	Pass@1	Pass@5	Pass@10
LLaMA-3-8B					
LoRA	0.52%	$62.8_{\pm 1.3}$	$36.7_{\pm 2.3}$	$45.6_{\pm 1.7}$	$48.3_{\pm 1.6}$
LoRA-RITE	0.52%	$64.6_{\pm 1.6}$	$38.2_{\pm 2.0}$	$46.8_{\pm 1.5}$	$48.9_{\pm 1.1}$
LORO	0.52%	$61.5_{\pm 1.2}$	$35.4_{\pm 2.3}$	$43.7_{\pm 1.5}$	$47.1_{\pm 1.1}$
RAdamW	0.52%	$63.9_{\pm 1.4}$	$38.3_{\pm 1.2}$	$47.1_{\pm 1.6}$	$50.2_{\pm 1.3}$
Riemannion	0.52%	$62.5_{\pm 1.3}$	$39.5_{\pm 0.7}$	$48.2_{\pm 1.3}$	$50.1_{\pm 0.5}$
Temper	0.52%	$\mathbf{65.5}_{\pm 1.5}$	$\mathbf{41.8}_{\pm 1.4}$	$\mathbf{50.2}_{\pm 0.7}$	$\mathbf{53.4}_{\pm 0.5}$
Qwen-2.5-7B					
LoRA	0.53%	$71.1_{\pm 1.3}$	$49.2_{\pm 0.9}$	$60.5_{\pm 1.5}$	$63.6_{\pm 0.8}$
LoRA-RITE	0.53%	$73.2_{\pm 1.4}$	$50.8_{\pm 2.2}$	$61.4_{\pm 0.8}$	$64.7_{\pm 0.7}$
LORO	0.53%	$68.7_{\pm 2.4}$	$48.4_{\pm 2.0}$	$56.2_{\pm 1.1}$	$57.3_{\pm 0.9}$
RAdamW	0.53%	$72.8_{\pm 0.5}$	$49.2_{\pm 2.1}$	$59.8_{\pm 1.4}$	$63.7_{\pm 1.0}$
Riemannion	0.53%	$70.6_{\pm 1.6}$	$50.3_{\pm 2.3}$	$61.1_{\pm 1.3}$	$62.5_{\pm 0.8}$
Temper	0.53%	$\mathbf{75.2}_{\pm 0.9}$	$\mathbf{52.6}_{\pm 2.1}$	$\mathbf{63.8}_{\pm 1.0}$	$\mathbf{66.4}_{\pm 0.7}$

4 EXPERIMENTS

Experiments Setup. We fine-tune Llama-3-8B (Grattafiori et al., 2024) and Qwen-2.5-7B (Yang et al., 2024). Every optimizer trains the same parameters: LoRA of rank $r = 16$ on all linear projections of every transformer block. The methods differ only in the update rule applied to those parameters. Commonsense reasoning is trained on Commonsense-170K (Hu et al., 2023) training mixture and evaluated on the eight benchmarks: BoolQ (Clark et al., 2019), PIQA (Bisk et al., 2020), SIQA (Sap et al., 2019), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2020), ARC-easy (Clark et al., 2018) and ARC-challenge (Clark et al., 2018), and OpenBookQA (Mihaylov et al., 2018). Math reasoning is trained on GSM8K (Cobbe et al., 2021) training set and evaluated by generative accuracy on the GSM8K test set. Code generation is trained on CodeAlpaca Chaudhary (2023) and evaluated on the HumanEval (Chen et al., 2021), reported as Pass@1, Pass@5 and Pass@10 over 20 samples per problem. We compare against LoRA trained with AdamW (Loshchilov and Hutter, 2019), LoRA-RITE (Yen et al., 2025), LORO (Mo et al., 2025), RAdamW (Bian et al., 2025) and Riemannion (Bogachev et al., 2026). These are the four invariant LoRA optimizers of Table 1 plus the standard baseline. Every optimizer is swept over its own learning-rate grid and we report its best setting; Appendix E gives the hyperparameter details.

Table 4: Every row is Temper with only the source of ρ and γ in (10) changed.

Statistic read from	# scales	Commonsense Avg.	GSM8K	Trained ppl
None (no second moment)	0	83.1	61.7	1.479
Factor gradients $\nabla B,\ \nabla A$	$m{+}n{+}2r$	83.6	62.4	1.415
Unmagnified gradients $\bar{\nabla} B,\ \bar{\nabla} A$	$m{+}n{+}2r$	85.1	63.6	1.366
Euclidean gradient ∇Z	$m{+}n$	84.2	62.7	1.403
G **(Temper)**	$m{+}n$	86.9	65.5	1.261

4.1 Main results

Temper is first on 23 of the 24 task–model cells of Tables 2 and 3. On the commonsense average it reaches 86.9 against 85.7 for LoRA-RITE on Llama-3-8B and 88.8 against 87.9 on Qwen-2.5-7B; the one exception is BoolQ on Qwen-2.5-7B, where LoRA-RITE reaches 75.0 against our 74.8. That suite leaves little room, with every optimizer between 85 and 96 on six of its eight tasks. The separations on math and code are larger: 0.9 and 2.0 points on GSM8K, and between 1.7 and 3.2 points at HumanEval Pass@1, Pass@5 and Pass@10. Six of those eight gaps exceed the combined seed spread of the two runs compared, against two of sixteen on the commonsense suite. Invariance alone does not account for the ordering. Riemannion is invariant and accumulates no adaptive statistic (Table 1), and it finishes below plain LoRA with AdamW, which is not invariant, on the commonsense average for both models. The three optimizers that do accumulate a second moment, LoRA-RITE, RAdamW and Temper, take the top three places on both. Both requirements are therefore doing work: Section 2.1 asks that the update be invariant and Section 2.2 that it carry adaptive scales. Among the three that meet both, the remaining difference is where the statistic is read: an $r \times r$ matrix on the unmagnified factor gradients, row and column sums of ∇Z, or row and column sums of G.

4.2 Ablation Analysis

Second moment. Table 4 runs Temper five times on Llama-3-8B, changing only the matrix (10) reads from. The commonsense average, GSM8K accuracy and the trained perplexity rank the five rows identically. Carrying a second moment at all is worth 3.8 points on the commonsense average; the best alternative source recovers 2.0 of them and the Euclidean gradient 1.1. Invariance accounts for part of the rest, since removing the magnification gains 1.5 points at an unchanged number of scales, but not for all of it: rows three and four are both invariant and still fall 1.8 and 2.7 points short of row five. The Euclidean gradient is also the more expensive of the two, because the factor gradients do not determine it (Proposition 3), so row four runs at $O(mnr)$ behind a backward hook against $O((m{+}n)r^2)$ for row three. The number of scales does not explain the ordering either, since the two best rows are the two with fewest.

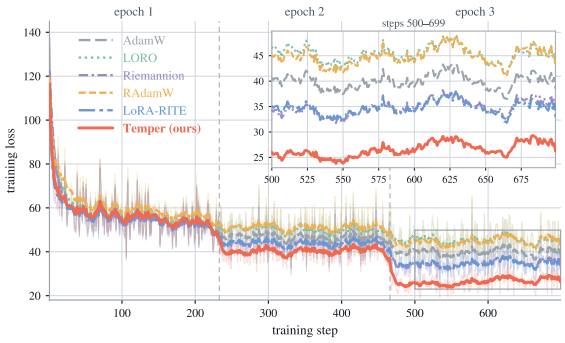

Figure 3: Training loss on GSM8K with Llama-3-8B. Thin lines are the raw loss, bold lines an exponential moving average; dashed verticals mark the epoch boundaries.

Training loss. Figure 3 records the comparison during optimization rather than at evaluation. Temper is below every baseline from within the first epoch and stays below through both epoch boundaries, so the separation is not produced by one epoch or by the end of the schedule. Every optimizer steps down at each boundary and the ordering survives both. The inset over the last 200 steps keeps Temper clear of the rest, with LoRA-RITE next and the two optimizers that carry no adaptive statistic, LORO and Riemannion, in the upper band.

4.3 Efficiency Analysis

Temper's state is 191.3 MiB, $1.03\times$ that of LoRA's Adam moments, because the two accumulators it adds hold $m + n$ numbers per layer rather than $mr + nr$. RAdamW keeps a dense $m \times n$ first moment and needs 5480 MiB, $28.7\times$ more, which is what forces its peak reserved memory to 16.63 GB and its throughput to 0.25 examples per second. The gap in peak reserved is only $1.96\times$ because the frozen base model and the activations (6.08 GB, common to every arm) dilute it; the state column is the direct measurement of the $O(mr+nr)$ claim of Section 2.2.

Table 5: Cost per optimizer at rank 16 on Llama-3.2-3B.

Method	State(MB)	Peak(GB)	Ex/s
LoRA	185.5	8.56	2.06
LoRA-RITE	185.9	8.57	1.17
LORO	371.0	8.67	1.28
RAdamW	5480	16.63	0.25
Riemannion	185.5	8.49	1.16
Temper	191.3	8.49	1.22

5 Related Work

Parameter-efficient fine-tuning. LoRA (Hu et al., 2022), AdaLoRA Zhang et al. (2023) and PiSSA Meng et al. (2024) add a low-rank term to a frozen weight, and merges it afterwards at no inference cost. Most work since varies what is parameterized rather than how it is trained: Liu et al. (2024) separate each column's magnitude from its direction, Huang et al. (2025) take a Hadamard product with W to escape the rank-r limit, Paischer et al. (2025) initialize from the directions of largest activation variance, and Kang and Yin (2026) share one trainable matrix across shards of W. All fix a parameterization and leave the update rule to a general-purpose optimizer; we take the parameterization as given and ask what that optimizer should be.

LoRA optimizers. Two constructions answer that. The first stays in factor coordinates: different learning rates for the two factors (Hayou et al., 2024), gradients adjusted so the update to Z approximates full fine-tuning (Wang et al., 2025), each factor gradient preconditioned by the inner product of the other (Zhang and Pilanci, 2024), one factor constrained to the Stiefel manifold (Park et al., 2025). All reduce the effect of an unbalanced factorization without removing it. Yen et al. (2025) prove that no diagonal preconditioner on the factor gradients can be invariant and answer with an $r \times r$ matrix on the unmagnified gradients; Temper does not contradict that result, because the map from the factor gradients to G is not diagonal (Section 3.3), and Yu et al. (2025) report the same property lost once a second moment is added. The second construction leaves the factors for the fixed-rank manifold of Vandereycken (2013), whose projection, retraction and transport we use. Bogachev et al. (2026) and Cesista et al. (2026) orthogonalize the direction (Jordan et al., 2024), invariant but carrying no adaptive scales; Mo et al. (2025) step along G but apply Adam to the factor gradients first. Bian et al. (2025) and Liu et al. (2026) do carry $m+n$ scales, but read them from ∇Z, which no rank-r step can follow (Section 2.2). Accumulating a second moment on G is older than this literature and Temper does not claim it: Bécigneul and Ganea (2019) give Riemannian Adam on product manifolds and Kasai et al. (2019) accumulate the same two marginals before projecting, both forming G as an $m \times n$ object. Temper adds the identities that avoid forming it, and takes Adafactor's rank-one reconstruction and exponent (Shazeer and Stern, 2018) rather than Shampoo's (Gupta et al., 2018). Table 1 places these optimizers by the statistic each accumulates and the object it is read from.

6 Conclusion

Transformation invariance settles which direction a LoRA step takes, not how far it moves along each coordinate. That is what a second moment decides, and the methods that step along the Riemannian gradient either carry none or read theirs from the Euclidean gradient, almost all of which lies where no rank-r step can follow. Temper reads it from the Riemannian gradient itself. Its rows and columns are the layer's outputs and inputs, which no refactorization moves, so the statistic is invariant although elementwise, and both marginals follow exactly from two $2r \times 2r$ Gram matrices, with no $m \times n$ matrix formed. Temper is therefore provably invariant and costs no more per step than an unpreconditioned one. On Llama-3-8B and Qwen-2.5-7B it leads the strongest baseline by 0.9 to 3.2 points. The scale and the direction it multiplies have to come from the same matrix, and on the fixed-rank manifold that matrix is the Riemannian gradient.

USAGE OF LLMS

In this work, LLMs are used for programming assistance and writing support. Specifically, they are employed to generate plotting and analysis scripts, debug existing code, and check derivations numerically, which substantially accelerates the development process. For writing support, they are mainly used to proofread the text and format *LaTeX* code.

ETHICS STATEMENT

This work proposes an optimizer for parameter-efficient fine-tuning. It collects no data and involves no human subjects; all experiments use public benchmarks and openly released base models. An optimizer is general-purpose and inherits the risks of whatever it is used to train: it makes fine-tuning cheaper and more effective without changing what a model is fine-tuned to do, so it offers no safeguard against misuse of the resulting models and none is claimed.

REPRODUCIBILITY STATEMENT

Section 3 gives the update rule in full, Appendix C writes it out operation by operation, and Appendix D accounts for its time and memory. Appendix E gives the base models, the adapter and its initialization, the per-task training protocol, the settings of every optimizer and the learning-rate grid each was swept on.

REFERENCES

Gary Bécigneul and Octavian-Eugen Ganea. Riemannian adaptive optimization methods. In *International Conference on Learning Representations (ICLR)*, 2019. arXiv:1810.00760.

Fengmiao Bian, Jinyang Zheng, Ziyun Liu, Jianzhou Luo, and Jian-Feng Cai. Finding low-rank matrix weights in DNNs via Riemannian optimization: RAdaGrad and RAdamW. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, et al. PIQA: reasoning about physical commonsense in natural language. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 7432–7439. AAAI Press, 2020. doi: 10.1609/AAAI.V34I05.6239. URL https://doi.org/10.1609/aaai.v34i05.6239.

Vladimir Bogachev, Vladimir Aletov, Alexander Molozhavenko, Denis Bobkov, Vera Soboleva, Aibek Alanov, and Maxim Rakhuba. LoRA meets Riemannion: Muon optimizer for parametrization-independent low-rank adapters. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2507.12142.

Franz Louis Cesista, Katherine Crowson, Cédric Simal, and Stella Biderman. LoRA-Muon: Spectral steepest descent on the low-rank manifold. *arXiv preprint arXiv:2606.12921*, 2026.

Sahil Chaudhary. Code alpaca: An instruction-following llama model for code generation, 2023.

Mark Chen, Jerry Tworek, Heewoo Jun, et al. Evaluating large language models trained on code. *CoRR*, abs/2107.03374, 2021. URL https://arxiv.org/abs/2107.03374.

Christopher Clark, Kenton Lee, Ming-Wei Chang, et al. Boolq: Exploring the surprising difficulty of natural yes/no questions. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 2924–2936. Association for Computational Linguistics, 2019. doi: 10.18653/V1/N19-1300. URL https://doi.org/10.18653/v1/n19-1300.

Peter Clark, Isaac Cowhey, Oren Etzioni, et al. Think you have solved question answering? try arc, the AI2 reasoning challenge. *CoRR*, abs/1803.05457, 2018. URL http://arxiv.org/abs/1803.05457.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Aaron Grattafiori, Abhimanyu Dubey, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization. In *International Conference on Machine Learning (ICML)*, 2018.

Soufiane Hayou, Nikhil Ghosh, and Bin Yu. LoRA+: Efficient low rank adaptation of large models. In *International Conference on Machine Learning (ICML)*, 2024. arXiv:2402.12354.

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

Zhiqiang Hu, Lei Wang, Yihuai Lan, et al. Llm-adapters: An adapter family for parameter-efficient fine-tuning of large language models. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 5254–5276. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.EMNLP-MAIN.319. URL https://doi.org/10.18653/v1/2023.emnlp-main.319.

Qiushi Huang, Tom Ko, Zhan Zhuang, Lilian Tang, and Yu Zhang. HiRA: Parameter-efficient Hadamard high-rank adaptation for large language models. In *International Conference on Learning Representations (ICLR)*, 2025.

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

Jiale Kang and Qingyu Yin. MiSS: Revisiting the trade-off in LoRA with an efficient shard-sharing structure. In *International Conference on Learning Representations (ICLR)*, 2026.

Hiroyuki Kasai, Pratik Jawanpuria, and Bamdev Mishra. Riemannian adaptive stochastic gradient algorithms on matrix manifolds. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 3262–3271. PMLR, 2019. URL http://proceedings.mlr.press/v97/kasai19a.html.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. DoRA: Weight-decomposed low-rank adaptation. In *International Conference on Machine Learning (ICML)*, 2024.

Ziyun Liu, Fengmiao Bian, and Jian-Feng Cai. AdaPreLoRA: Adafactor preconditioned low-rank adaptation. *arXiv preprint arXiv:2605.08734*, 2026.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019.

Fanxu Meng, Zhaohui Wang, and Muhan Zhang. PiSSA: Principal singular values and singular vectors adaptation of large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? A new dataset for open book question answering. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2381–2391. Association for Computational Linguistics, 2018. doi: 10.18653/V1/D18-1260. URL https://doi.org/10.18653/v1/d18-1260.

Zhanfeng Mo, Long-Kai Huang, and Sinno Jialin Pan. Parameter and memory efficient pretraining via low-rank Riemannian optimization. In *International Conference on Learning Representations (ICLR)*, 2025.

Fabian Paischer, Lukas Hauzenberger, Thomas Schmied, Benedikt Alkin, Marc Deisenroth, and Sepp Hochreiter. Parameter efficient fine-tuning via explained variance adaptation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Juneyoung Park, Minjae Kang, Seongbae Lee, Haegang Lee, Seongwan Kim, and Jaeho Lee. Riemannian optimization for LoRA on the Stiefel manifold. *arXiv preprint arXiv:2508.17901*, 2025.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8732–8740. AAAI Press, 2020. doi: 10.1609/AAAI.V34I05.6399. URL https://doi.org/10.1609/aaai.v34i05.6399.

Maarten Sap, Hannah Rashkin, Derek Chen, et al. Socialiqa: Commonsense reasoning about social interactions. *CoRR*, abs/1904.09728, 2019. URL http://arxiv.org/abs/1904.09728.

Noam Shazeer and Mitchell Stern. Adafactor: Adaptive learning rates with sublinear memory cost. In *International Conference on Machine Learning (ICML)*, 2018.

Bart Vandereycken. Low-rank matrix completion by Riemannian optimization. *SIAM Journal on Optimization*, 23(2):1214–1236, 2013.

Zhengbo Wang, Jian Liang, Ran He, Zilei Wang, and Tieniu Tan. Lora-pro: Are low-rank adapters properly optimized? In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=gTwRMU3lJ5.

An Yang, Baosong Yang, Binyuan Hui, et al. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024.

Jui-Nan Yen, Si Si, Zhao Meng, Felix X. Yu, Sai Surya Duvvuri, Inderjit S. Dhillon, Cho-Jui Hsieh, and Sanjiv Kumar. Lora done RITE: robust invariant transformation equilibration for lora optimization. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=VpWki1v2P8.

Xin Yu, Yujia Wang, Jinghui Chen, and Lingzhou Xue. AltLoRA: Towards better gradient approximation in low-rank adaptation with alternating projections. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. arXiv:2505.12455.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, et al. Hellaswag: Can a machine really finish your sentence? In Anna Korhonen, David R. Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 4791–4800. Association for Computational Linguistics, 2019. doi: 10.18653/V1/P19-1472. URL https://doi.org/10.18653/v1/p19-1472.

Fangzhao Zhang and Mert Pilanci. Riemannian preconditioned LoRA for fine-tuning foundation models. In *International Conference on Machine Learning (ICML)*, 2024. arXiv:2402.02347.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. AdaLoRA: Adaptive budget allocation for parameter-efficient fine-tuning. In *International Conference on Learning Representations (ICLR)*, 2023.

A Deferred proofs and derivations

This appendix is self-contained. We restate here everything the proofs use, so that none of them requires turning back to the main text.

Table 6: Notation used in this appendix. Throughout, B and A have full rank r.

Symbol	Meaning
$Z = BA$	the adapted weight, with $B \in \mathbb{R}^{m\times r}$ and $A \in \mathbb{R}^{r\times n}$
$\nabla Z,\ \nabla B,\ \nabla A$	gradients of the loss; the chain rule gives $\nabla B = \nabla Z\, A^\top$ and $\nabla A = B^\top \nabla Z$
$B = U_B R_B,\ \ A^\top = U_A R_A$	polar decompositions, with $U_B^\top U_B = U_A^\top U_A = I_r$ and R_B, R_A symmetric positive definite
$\Pi = U_B U_B^\top,\ \ \Lambda = U_A U_A^\top$	the orthogonal projectors onto $\mathrm{col}(B)$ and $\mathrm{col}(A^\top)$; $\Pi^2 = \Pi = \Pi^\top$ and likewise for Λ
$\bar\nabla B = \nabla Z\, U_A,\ \ \bar\nabla A = U_B^\top \nabla Z$	the unmagnified gradients; equivalently $\nabla B\, R_A^\dagger$ and $R_B^\dagger\, \nabla A$
$P_Z(Y) = \Pi\, Y + (I-\Pi)\, Y\, \Lambda$	the orthogonal projection onto the tangent space of $\mathcal{M}_r$ at Z, the manifold of rank-r matrices
$G = P_Z(\nabla Z),\ \ N = \nabla Z - G$	the Riemannian gradient and the normal component
$G = \dot B\, U_A^\top + U_B\, \dot A = PQ^\top$	its factored form, with $\dot B = (I-\Pi)\bar\nabla B$, $\dot A = \bar\nabla A$, $P = [\,\dot B,\, U_B\,] \in \mathbb{R}^{m\times 2r}$ and $Q = [\,U_A,\, \dot A^\top\,] \in \mathbb{R}^{n\times 2r}$
$\rho(X)_i,\ \ \gamma(X)_j$	the row and column sums of squares of X, that is $\sum_j X_{ij}^2$ and $\sum_i X_{ij}^2$
$(B, A) \mapsto (BR,\ R^{-1}A)$	the transformation of Definition 1, for any invertible $R \in \mathbb{R}^{r\times r}$; it leaves Z unchanged

A.1 Two facts about the projection

Several proofs below use the same two properties of P_Z, so we record them once.

Lemma 1. *P_Z is idempotent and self-adjoint for the Frobenius inner product $\langle X, Y\rangle_F = \mathrm{tr}(X^\top Y)$. Consequently, for every Y the decomposition $Y = P_Z(Y) + \big(Y - P_Z(Y)\big)$ is orthogonal, and $\|Y\|_F^2 = \|P_Z(Y)\|_F^2 + \|Y - P_Z(Y)\|_F^2$.*

Proof. Write $W = P_Z(Y) = \Pi Y + (I-\Pi)Y\Lambda$. Since $\Pi(I-\Pi) = \Pi - \Pi^2 = 0$,

$$\Pi W = \Pi^2 Y + \Pi(I-\Pi)Y\Lambda = \Pi Y,$$

and since $(I-\Pi)^2 = I - \Pi$ and $\Lambda^2 = \Lambda$,

$$(I-\Pi)W\Lambda = (I-\Pi)\Pi Y\Lambda + (I-\Pi)^2 Y\Lambda^2 = 0 + (I-\Pi)Y\Lambda.$$

Adding the two lines gives $P_Z(W) = \Pi Y + (I-\Pi)Y\Lambda = W$, so $P_Z \circ P_Z = P_Z$. For self-adjointness, use $\langle \Pi X, Y\rangle_F = \langle X, \Pi Y\rangle_F$ and $\langle XY, Z\rangle_F = \langle X, ZY^\top\rangle_F$ together with $\Pi^\top = \Pi$ and $\Lambda^\top = \Lambda$:

$$\begin{aligned}\langle P_Z(X), Y\rangle_F &= \langle \Pi X, Y\rangle_F + \langle (I-\Pi)X\Lambda, Y\rangle_F \\ &= \langle X, \Pi Y\rangle_F + \langle X, (I-\Pi)Y\Lambda\rangle_F = \langle X, P_Z(Y)\rangle_F.\end{aligned}$$

An idempotent self-adjoint map is an orthogonal projection, which gives the decomposition and the Pythagorean identity. □

A.2 THE ROW-SCALE COUNT

Section 2.2 claims that one learning rate needs a number of steps proportional to the spread of the curvature across rows, and that one scale per row removes that dependence. Here is the statement in full.

Take a loss that separates over the rows of Z with unequal curvatures,

$$f(Z) = \tfrac{1}{2}\sum_{i=1}^{m} w_i \left\| Z_{i,:} - Z^\star_{i,:} \right\|_2^2, \qquad 0 < w_{\min} \le w_i \le w_{\max}, \qquad \kappa := \frac{w_{\max}}{w_{\min}}, \tag{18}$$

whose gradient is $\nabla Z_{i,:} = w_i\,(Z_{i,:} - Z^\star_{i,:})$, one row at a time. Write $e_i^{(t)} := Z_{i,:}^{(t)} - Z^\star_{i,:}$ for the error in row i after t steps.

One learning rate. The step $\delta Z = -\eta\,\nabla Z$ gives

$$e_i^{(t+1)} = e_i^{(t)} - \eta\, w_i\, e_i^{(t)} = (1-\eta w_i)\, e_i^{(t)}, \qquad \text{so} \qquad \left\| e_i^{(t)} \right\|_2 = |1-\eta w_i|^{\,t} \left\| e_i^{(0)} \right\|_2. \tag{19}$$

Every row must contract, so the relevant quantity is $\max_i |1-\eta w_i|$, and because $\eta \mapsto |1-\eta w|$ is decreasing in w below $1/\eta$ and increasing above it, the maximum over $w \in [w_{\min}, w_{\max}]$ is attained at one of the two endpoints:

$$\max_i |1-\eta w_i| \;\le\; \max\left\{\, |1-\eta w_{\min}|,\; |1-\eta w_{\max}| \,\right\}. \tag{20}$$

The right-hand side is minimized where the two terms are equal, that is where $1-\eta w_{\min} = \eta w_{\max} - 1$, giving

$$\eta^\star = \frac{2}{w_{\min} + w_{\max}}, \qquad \max_i |1-\eta^\star w_i| \;=\; \frac{w_{\max} - w_{\min}}{w_{\max} + w_{\min}} \;=\; \frac{\kappa - 1}{\kappa + 1}. \tag{21}$$

Reaching a relative error ϵ on every row therefore needs

$$t \;\ge\; \frac{\log(1/\epsilon)}{\log \frac{\kappa+1}{\kappa-1}}, \qquad \log\frac{\kappa+1}{\kappa-1} = 2\,\operatorname{artanh}\frac{1}{\kappa} = \frac{2}{\kappa} + O(\kappa^{-3}), \tag{22}$$

so $t \approx \frac{\kappa}{2}\log(1/\epsilon)$ steps: the count grows linearly in the spread of the curvature.

One scale per row. Let the step be $\delta Z_{i,:} = -\eta\, d_i\, \nabla Z_{i,:}$ with one positive number d_i per row. Then $e_i^{(t+1)} = (1-\eta d_i w_i) e_i^{(t)}$, and the choice $d_i = 1/w_i$ makes every factor equal to $1-\eta$ at once. The count becomes $\log(1/\epsilon)/\log\frac{1}{1-\eta}$, with no κ in it. The m numbers d_i are exactly what (20) could not supply with a single η.

Two remarks connect this to the rest of the paper. First, the quantity $\mu_i(\delta Z) = \|\delta Z_{i,:}\|_2 / \|Z_{i,:}\|_2$ of Section 2.2 reads the same imbalance off the update rather than off the loss, which is what can be measured when w_i is not available. Second, an adaptive optimizer does not know w_i either and estimates d_i from a running average of squared gradients; under (18) that average is proportional to $w_i^2\|e_i\|^2$, so its inverse square root stands in for $1/w_i$ up to the factor $\|e_i\|$.

A.3 INVARIANCE OF THE RIEMANNIAN GRADIENT

Proposition 1. *G does not change under the transformation of Table 6. Every function of G read as an $m \times n$ matrix is therefore invariant, elementwise functions included.*

Proof. Let $(B', A') = (BR, R^{-1}A)$. Since R is invertible,

$$\operatorname{col}(B') = \operatorname{col}(BR) = \operatorname{col}(B), \qquad \operatorname{col}(A'^\top) = \operatorname{col}(A^\top R^{-\top}) = \operatorname{col}(A^\top).$$

The orthogonal projector onto a subspace is determined by that subspace alone, so the two projectors of Table 6 satisfy $\Pi' = \Pi$ and $\Lambda' = \Lambda$, even though the individual frames U_B and U_A do change. Next, $Z' = B'A' = BRR^{-1}A = BA = Z$, and ∇Z is a function of Z and the data, so $\nabla Z' = \nabla Z$. Since P_Z involves the factors only through Π and Λ,

$$G' = \Pi'\,\nabla Z' + (I - \Pi')\,\nabla Z'\,\Lambda' = \Pi\,\nabla Z + (I-\Pi)\,\nabla Z\,\Lambda = G.$$

If $G' = G$ as matrices then $h(G') = h(G)$ for any function h of the matrix, in particular for one applied entry by entry. □

The last sentence is the reason an elementwise second moment is admissible here and is not admissible on the factor gradients: ∇B and ∇A carry R, whereas G does not.

A.4 What the factor gradients see

Proposition 3. *$\nabla B = GA^\top$ and $\nabla A = B^\top G$ hold exactly. Writing $\nabla Z = G + N$ with $N := \nabla Z - P_Z(\nabla Z)$, the normal component satisfies $NA^\top = 0$ and $B^\top N = 0$.*

Proof. We first record two identities. From $A^\top = U_A R_A$ and $U_A^\top U_A = I_r$,

$$\Lambda A^\top = U_A U_A^\top U_A R_A = U_A \left(U_A^\top U_A\right) R_A = U_A R_A = A^\top, \tag{23}$$

and from $B = U_B R_B$ with R_B symmetric,

$$B^\top \Pi = R_B U_B^\top U_B U_B^\top = R_B \left(U_B^\top U_B\right) U_B^\top = R_B U_B^\top = B^\top, \qquad \text{hence} \qquad B^\top (I - \Pi) = 0. \tag{24}$$

Now expand $GA^\top$ and then (23):

$$\begin{aligned} GA^\top &= \Pi \nabla Z\, A^\top + (I - \Pi) \nabla Z\, \Lambda A^\top \\ &= \Pi \nabla Z\, A^\top + (I - \Pi) \nabla Z\, A^\top \\ &= \big(\Pi + I - \Pi\big) \nabla Z\, A^\top = \nabla Z\, A^\top = \nabla B, \end{aligned}$$

the last equality. Symmetrically, using (24),

$$B^\top G = B^\top \Pi \nabla Z + \underbrace{B^\top (I - \Pi)}_{=0} \nabla Z\, \Lambda = B^\top \nabla Z = \nabla A.$$

For the normal component, subtract: $N = \nabla Z - G$, so

$$NA^\top = \nabla Z A^\top - GA^\top = \nabla B - \nabla B = 0, \qquad B^\top N = B^\top \nabla Z - B^\top G = \nabla A - \nabla A = 0. \ \square$$

Two consequences are used in the main text. Replacing ∇Z by G in the chain rule changes neither factor gradient, so an optimizer that reads only ∇B and ∇A cannot distinguish the two; and the map from the factor gradients to ∇Z is not invertible, so ∇Z cannot be reconstructed from them and has to be captured separately.

A.5 What the two scales do to the rows

Proposition 2. *Let $\rho_i := \rho(G)_i > 0$ and $\gamma_j := \gamma(G)_j > 0$, and let*

$$Y := D^{\mathrm{row}}\, G\, D^{\mathrm{col}}, \qquad D^{\mathrm{row}} = \operatorname{diag}\left(\rho^{-1/2}\right), \qquad D^{\mathrm{col}} = \operatorname{diag}\left(\gamma^{-1/2}\right). \tag{25}$$

Then every row of Y satisfies

$$\frac{1}{\max_j \gamma_j} \;\le\; \|Y_{i,:}\|_2^2 \;\le\; \frac{1}{\min_j \gamma_j}, \qquad \text{hence} \qquad \frac{\max_i \|Y_{i,:}\|_2}{\min_i \|Y_{i,:}\|_2} \;\le\; \left(\frac{\max_j \gamma_j}{\min_j \gamma_j}\right)^{1/2}. \tag{26}$$

Neither bound involves ρ.

Proof. Both D^{row} and D^{col} are diagonal, so (25) is entrywise

$$Y_{ij} = \rho_i^{-1/2}\, G_{ij}\, \gamma_j^{-1/2}.$$

Square and sum along row i:

$$\|Y_{i,:}\|_2^2 = \sum_{j=1}^{n} \frac{G_{ij}^2}{\rho_i\, \gamma_j} = \sum_{j=1}^{n} w_{ij}\, \gamma_j^{-1}, \qquad w_{ij} := \frac{G_{ij}^2}{\rho_i}. \tag{27}$$

The weights are non-negative, and they sum to one along the row because ρ_i is the row sum of the squared entries of the same matrix:

$$\sum_{j=1}^{n} w_{ij} = \frac{1}{\rho_i} \sum_{j=1}^{n} G_{ij}^2 = \frac{\rho_i}{\rho_i} = 1. \tag{28}$$

So (27) is a convex combination of the n numbers $\gamma_1^{-1}, \ldots, \gamma_n^{-1}$, and a convex combination lies between the smallest and the largest of them:

$$\min_j \gamma_j^{-1} \;\le\; \|Y_{i,:}\|_2^2 \;\le\; \max_j \gamma_j^{-1}, \qquad \text{that is} \qquad \frac{1}{\max_j \gamma_j} \;\le\; \|Y_{i,:}\|_2^2 \;\le\; \frac{1}{\min_j \gamma_j}.$$

The bounds do not depend on i. Taking square roots and dividing the largest by the smallest gives the second half of (26). $\square$

The content of the proof is (28). Dividing row i by $\rho_i^{1/2}$ makes the row's own weights sum to one, so the row profile of G is removed at that step and what remains is the column profile, a different quantity.

A.6 Reading the two vectors from the wrong matrix

Equation (28) holds only because ρ_i is a statistic of the same matrix the scales multiply. The following keeps the form of (25) and changes only where the two vectors are read from, which is the comparison the ablation of Section 4.2 runs.

Proposition 4. *Let $\rho' := \rho(\nabla Z)$ and $\gamma' := \gamma(\nabla Z)$ be read from the Euclidean gradient while the matrix being scaled is still G, and set $Y' := \operatorname{diag}(\rho'^{-1/2})\, G\, \operatorname{diag}(\gamma'^{-1/2})$. Put $\kappa_i := \rho(G)_i/\rho'_i$. Then*

$$\|Y'_{i,:}\|_2^2 = \kappa_i \sum_{j=1}^{n} w_{ij}\, \gamma_j'^{-1}, \qquad \frac{\kappa_i}{\max_j \gamma'_j} \;\le\; \|Y'_{i,:}\|_2^2 \;\le\; \frac{\kappa_i}{\min_j \gamma'_j}, \tag{29}$$

with w_{ij} as in (27)*, and therefore*

$$\frac{\max_i \|Y'_{i,:}\|_2}{\min_i \|Y'_{i,:}\|_2} \;\le\; \left(\frac{\kappa_{\max}}{\kappa_{\min}} \cdot \frac{\max_j \gamma'_j}{\min_j \gamma'_j}\right)^{1/2}. \tag{30}$$

Proof. Entrywise $Y'_{ij} = \rho_i'^{-1/2} G_{ij} \gamma_j'^{-1/2}$, so

$$\|Y'_{i,:}\|_2^2 = \sum_{j=1}^{n} \frac{G_{ij}^2}{\rho'_i\, \gamma'_j} = \frac{\rho(G)_i}{\rho'_i} \sum_{j=1}^{n} \frac{G_{ij}^2}{\rho(G)_i\, \gamma'_j} = \kappa_i \sum_{j=1}^{n} w_{ij}\, \gamma_j'^{-1},$$

where the middle step multiplies and divides by $\rho(G)_i$. The weights w_{ij} still sum to one by (28), so the sum is again a convex combination and lies between $1/\max_j \gamma'_j$ and $1/\min_j \gamma'_j$; multiplying by $\kappa_i > 0$ gives the two bounds in (29). Dividing the largest possible value of $\|Y'_{i,:}\|_2^2$ by the smallest and taking a square root gives (30). □

Comparing (30) with (26): the bound has acquired the factor $(\kappa_{\max}/\kappa_{\min})^{1/2}$, and $\kappa_i = \rho(G)_i/\rho(\nabla Z)_i$ is the row profile of G measured against that of ∇Z. Proposition 2 removes that profile; reading the scales from ∇Z lets it back in. The exponent and the projection are the same in both statements, so the only thing that differs is the source of the two vectors.

A.7 How far the projection moves a row

Proposition 5. *Let $F = P_Z(Y)$ for any $Y \in \mathbb{R}^{m\times n}$. Then*

$$\|F_{i,:}\|_2 \;\le\; \|Y_{i,:}\|_2 \;+\; \Pi_{ii}^{1/2}\, \sigma_{\max}(Y), \tag{31}$$

where $\Pi_{ii} = \|(U_B)_{i,:}\|_2^2$ is the leverage score of row i in $\operatorname{col}(B)$ and $\sum_{i=1}^{m} \Pi_{ii} = r$.

Proof. Rewrite P_Z by adding and subtracting $\Pi Y \Lambda$:

$$P_Z(Y) = \Pi Y + Y\Lambda - \Pi Y \Lambda = \underbrace{\Pi\, Y\, (I-\Lambda)}_{\text{(a)}} \;+\; \underbrace{Y\,\Lambda}_{\text{(b)}}. \tag{32}$$

Take row i of each term and apply the triangle inequality.

For (b), row i is $Y_{i,:}\Lambda$, and Λ is an orthogonal projector, so $\|Y_{i,:}\Lambda\|_2 \le \|Y_{i,:}\|_2$.

For (a), write row i as $e_i^\top \Pi\, Y(I-\Lambda)$ with e_i the i-th standard basis vector. For any row vector $u^\top$ and matrix X we have $\|u^\top X\|_2 \le \|u\|_2\, \sigma_{\max}(X)$, so

$$\big\|e_i^\top \Pi\, Y(I-\Lambda)\big\|_2 \;\le\; \|\Pi e_i\|_2 \cdot \sigma_{\max}\big(Y(I-\Lambda)\big) \;\le\; \|\Pi e_i\|_2 \cdot \sigma_{\max}(Y),$$

the last step because $I - \Lambda$ is an orthogonal projector and therefore has all singular values in $\{0, 1\}$. The remaining factor is the leverage score, since Π is idempotent and symmetric:

$$\|\Pi e_i\|_2^2 = e_i^\top \Pi^\top \Pi e_i = e_i^\top \Pi e_i = \Pi_{ii} = \left(U_B U_B^\top\right)_{ii} = \|(U_B)_{i,:}\|_2^2.$$

Adding the two bounds gives (31). Finally, by the cyclic property of the trace and $U_B^\top U_B = I_r$,

$$\sum_{i=1}^{m} \Pi_{ii} = \operatorname{tr}\left(U_B U_B^\top\right) = \operatorname{tr}\left(U_B^\top U_B\right) = \operatorname{tr}(I_r) = r. \qquad \square$$

Since the m leverage scores sum to r, their average is r/m, so the projection can move only a few rows appreciably: at most r/δ rows have $\Pi_{ii} \geq \delta$.

A.8 The transport of the first moment

The first moment is stored as a tangent vector at the previous iterate and has to be brought to the current one before it can be added to G_t. Temper uses the projection as the transport. This subsection shows that the operation runs on the stored pair and never builds an $m \times n$ matrix.

Write $\tilde{U}_B, \tilde{U}_A$ for the frames of step $t-1$, and let the stored moment be

$$M_{t-1} = \dot{M}_B \, \tilde{U}_A^\top + \tilde{U}_B \, \dot{M}_A, \qquad \dot{M}_B \in \mathbb{R}^{m \times r}, \quad \dot{M}_A \in \mathbb{R}^{r \times n}, \qquad \tilde{U}_B^\top \dot{M}_B = 0, \tag{33}$$

which is the factored form at Z_{t-1}; the condition $\tilde{U}_B^\top \dot{M}_B = 0$ makes the pair $(\dot{M}_B, \dot{M}_A)$ unique.

Proposition 6. *With $\Pi = U_B U_B^\top$ and $\Lambda = U_A U_A^\top$ the projectors at Z_t, define the four $r \times r$ matrices*

$$\Theta_A := \tilde{U}_A^\top U_A, \qquad \Theta_B := U_B^\top \tilde{U}_B, \qquad \alpha := U_B^\top \dot{M}_B, \qquad \beta := \dot{M}_A \, U_A. \tag{34}$$

Then $P_{Z_t}(M_{t-1}) = \dot{M}'_B \, U_A^\top + U_B \, \dot{M}'_A$ with

$$\dot{M}'_B = (I - \Pi)\big[\, \dot{M}_B \, \Theta_A + \tilde{U}_B \, \beta \,\big], \qquad \dot{M}'_A = \alpha \, \tilde{U}_A^\top + \Theta_B \, \dot{M}_A, \tag{35}$$

and $U_B^\top \dot{M}'_B = 0$. Every product in (34) *and* (35) *involves a matrix of size $m \times r$, $n \times r$ or $r \times r$, so the whole operation costs $O\big((m+n)r^2\big)$ and allocates nothing of size $m \times n$.*

Proof. Apply P_Z to $M := M_{t-1}$ and treat the two terms separately. For the first, substitute (33) and insert $U_B U_B^\top$:

$$\begin{aligned} \Pi M &= U_B U_B^\top \big(\dot{M}_B \tilde{U}_A^\top + \tilde{U}_B \dot{M}_A\big) \\ &= U_B \big[\, (U_B^\top \dot{M}_B)\, \tilde{U}_A^\top + (U_B^\top \tilde{U}_B)\, \dot{M}_A \,\big] = U_B \big[\, \alpha \, \tilde{U}_A^\top + \Theta_B \, \dot{M}_A \,\big] = U_B \, \dot{M}'_A. \end{aligned}$$

For the second, first compute $M\Lambda$:

$$\begin{aligned} M\Lambda &= \big(\dot{M}_B \tilde{U}_A^\top + \tilde{U}_B \dot{M}_A\big) U_A U_A^\top \\ &= \big[\, \dot{M}_B \, (\tilde{U}_A^\top U_A) + \tilde{U}_B \, (\dot{M}_A U_A) \,\big] U_A^\top = \big[\, \dot{M}_B \, \Theta_A + \tilde{U}_B \, \beta \,\big] \, U_A^\top, \end{aligned}$$

and then apply $I - \Pi$ on the left, which does not touch the trailing $U_A^\top$:

$$(I - \Pi)\, M\Lambda = \Big((I - \Pi)\big[\, \dot{M}_B \Theta_A + \tilde{U}_B \beta \,\big]\Big) U_A^\top = \dot{M}'_B \, U_A^\top.$$

Adding the two terms gives $P_{Z_t}(M_{t-1}) = \dot{M}'_B U_A^\top + U_B \dot{M}'_A$, which is the form (**??**) at Z_t. The orthogonality $U_B^\top \dot{M}'_B = 0$ holds because $\dot{M}'_B$ carries the factor $I - \Pi$ and $U_B^\top (I - \Pi) = U_B^\top - U_B^\top = 0$.

For the cost, Θ_A and α are inner products of two matrices with r columns and m or n rows, at $2mr^2$ and $2nr^2$ flops; Θ_B and β likewise. In (35), $\dot{M}_B \Theta_A$ and $\tilde{U}_B \beta$ are $m \times r$ by $r \times r$ products, the projection $I - \Pi$ costs two more such products, and $\alpha \tilde{U}_A^\top$ and $\Theta_B \dot{M}_A$ are $r \times r$ by $r \times n$ products. Summing, the operation is $12mr^2 + 8nr^2 + O\big((m+n)r\big)$, and no intermediate has both dimensions large. $\square$

A.9 THE NORMAL COMPONENT

Proposition 7. *Let $N = \nabla Z - P_Z(\nabla Z)$. Then $\langle G, N\rangle_F = 0$ and $\|\nabla Z\|_F^2 = \|G\|_F^2 + \|N\|_F^2$. The tangent space to $\mathcal{M}_r$ at a rank-r Z has dimension $r(m+n-r)$ and its orthogonal complement has dimension $(m-r)(n-r)$, so if ∇Z has independent entries of equal variance,*

$$\frac{\mathbb{E}\|N\|_F^2}{\mathbb{E}\|\nabla Z\|_F^2} = \frac{(m-r)(n-r)}{mn}. \tag{36}$$

Proof. The first claim is Lemma 1 applied to $Y = \nabla Z$.

For the dimension, use the parameterization factored form: a tangent vector at Z is $\dot{B}\,U_A^\top + U_B\,\dot{A}$ with $\dot{A} \in \mathbb{R}^{r\times n}$ free and $\dot{B} \in \mathbb{R}^{m\times r}$ constrained by $U_B^\top \dot{B} = 0$. The first block contributes rn free parameters. The second lives in the set of $m \times r$ matrices whose columns are orthogonal to the r-dimensional $\mathrm{col}(U_B)$, which has dimension $(m-r)r$. The map from the pair to the matrix is injective, because $U_B^\top\big(\dot{B}U_A^\top + U_B\dot{A}\big)U_A = \dot{A}U_A$ recovers one block and the other follows by subtraction. Hence

$$\dim T_Z\mathcal{M}_r = rn + (m-r)r = r(m+n-r),$$

and, the ambient space having dimension mn,

$$\dim\big(T_Z\mathcal{M}_r\big)^\perp = mn - r(m+n-r) = mn - rm - rn + r^2 = (m-r)(n-r).$$

Finally, let X have independent entries of variance σ^2 and let S be any subspace of $\mathbb{R}^{m\times n}$ with orthonormal basis $E_1, \dots, E_d$. Then $\mathbb{E}\|P_S X\|_F^2 = \sum_{k=1}^d \mathbb{E}\langle X, E_k\rangle_F^2 = d\,\sigma^2$, because each $\langle X, E_k\rangle_F$ is a unit-norm linear form in the entries of X. Applying this with $S = (T_Z\mathcal{M}_r)^\perp$ of dimension $(m-r)(n-r)$ and with $S = \mathbb{R}^{m\times n}$ of dimension mn gives (36). □

When $r \ll \min(m,n)$ the right-hand side of (36) is close to one: at $m = n = 1024$ and $r = 16$ it is 0.969. Together with Proposition 3, which says N reaches neither factor gradient, this is the sense in which most of the Euclidean gradient is unavailable to any rank-r step.

B PROOF OF THE INVARIANCE THEOREM

Theorem 1. *Assume the factors keep rank r along the trajectory and work in exact arithmetic. Then for any invertible R applied to the initial factors, the quantities G_t, V_t^{row}, V_t^{col}, $\hat{M}_t$ and F_t agree at every step, and so does $\{Z_t\}$.*

Fix an invertible $R \in \mathbb{R}^{r\times r}$. Let trajectory 1 start from (B_1, A_1) and trajectory 2 from $(B_2, A_2) = (B_1R,\ R^{-1}A_1)$, so that $Z_0 = B_1A_1 = B_2A_2$ is the same on both. Both run Algorithm 1 with the same learning rate, the same β_1, β_2 and ϵ, and the same data in the same order. We prove by induction on t that Z_t, M_{t-1}, V_{t-1}^{row} and V_{t-1}^{col} agree between the two.

Base case. Z_0 agrees by construction, and the state is initialized to zero on both trajectories, so $M_{-1} = 0$ and $V_{-1}^{\mathrm{row}} = V_{-1}^{\mathrm{col}} = 0$ agree.

Inductive step. Assume Z_t, M_{t-1}, V_{t-1}^{row} and V_{t-1}^{col} agree. We take the lines of Algorithm 1 in order.

Lines 1–3: the Riemannian gradient. ∇Z_t is a function of Z_t and the data, so it agrees. By Proposition 1, $G_t = P_{Z_t}(\nabla Z_t)$ agrees.

This is the one place where the full-rank assumption is used. Line 2 recovers the two unmagnified gradients as $\bar{\nabla}B = \nabla B\,R_A^\dagger$ and $\bar{\nabla}A = R_B^\dagger\,\nabla A$. When R_A and R_B are invertible, $R_A^\dagger = R_A^{-1}$ and $R_B^\dagger = R_B^{-1}$, and the definitions gives $\bar{\nabla}B = \nabla Z\,U_A$ and $\bar{\nabla}A = U_B^\top\nabla Z$, which depend on the factors only through U_A and U_B. If the cutoff in the pseudo-inverse is active, the recovered quantities are truncations that depend on which factorization is stored, and the argument breaks at this line and at no other.

Line 4: the two marginals. $\rho(G_t)$ and $\gamma(G_t)$ are functions of G_t read as an $m \times n$ matrix, so they agree, and so do V_t^{row}, V_t^{col} and the bias-corrected $\widehat{V}_t$.

One point deserves care, because the implementation does not compute the marginals from G_t but from P and Q, and neither of those is invariant on its own. The polar decomposition of $B' = BR$ is obtained by polar-decomposing the $r \times r$ matrix $R_B R$ as $R_B R = W_B S_B$ with W_B orthogonal and S_B symmetric positive definite, giving

$$B' = U_B R_B R = (U_B W_B)\, S_B, \qquad \text{so} \qquad U_{B'} = U_B W_B,$$

and symmetrically $A'^{\top} = A^{\top} R^{-\top} = U_A (R_A R^{-\top}) = (U_A W_A) S_A$, so $U_{A'} = U_A W_A$ with W_A orthogonal. The two unmagnified gradients inherit the rotations:

$$\bar{\nabla} B' = \nabla Z\, U_{A'} = \bar{\nabla} B\, W_A, \qquad \bar{\nabla} A' = U_{B'}^{\top} \nabla Z = W_B^{\top}\, \bar{\nabla} A,$$

and since $\Pi' = \Pi$ by Proposition 1, also $\dot{B}' = \dot{B}\, W_A$ and $\dot{A}' = W_B^{\top} \dot{A}$. Substituting into the factored form,

$$P \mapsto P\,\Theta, \qquad Q \mapsto Q\,\Theta, \qquad \Theta := \begin{bmatrix} W_A & 0 \\ 0 & W_B \end{bmatrix} \in \mathbb{R}^{2r \times 2r}, \qquad \Theta^{\top}\Theta = I_{2r}. \tag{37}$$

The same Θ appears in both factors, so $P'Q'^{\top} = P\Theta\Theta^{\top}Q^{\top} = PQ^{\top}$, as it must be since that product is G. The marginals are unchanged for the same reason:

$$\begin{aligned} \operatorname{diag}\left(P'(Q'^{\top}Q')P'^{\top}\right) &= \operatorname{diag}\left(P\Theta\,\Theta^{\top}Q^{\top}Q\Theta\,\Theta^{\top}P^{\top}\right) \\ &= \operatorname{diag}\left(P\,(Q^{\top}Q)\,P^{\top}\right) = \rho(G), \end{aligned}$$

and likewise $\operatorname{diag}\left(Q'(P'^{\top}P')Q'^{\top}\right) = \gamma(G)$. So the Gram route computes an invariant quantity from two non-invariant factors.

Line 5: the first moment. The update is $M_t = \beta_1 P_{Z_t}(M_{t-1}) + (1-\beta_1)G_t$. The map P_{Z_t} depends on Z_t alone and therefore agrees, M_{t-1} agrees by the induction hypothesis, and G_t agrees, so M_t and its bias-corrected form $\hat{M}_t$ agree.

Line 6: the step. $\widehat{V}_t$ and $\hat{M}_t$ agree, so the entrywise quotient $\hat{M}_t / \sqrt{\widehat{V}_t}$ agrees, and F_t agrees because P_{Z_t} does. Hence $Z_t - \eta F_t$ agrees.

Lines 7–8: the retraction. The truncated decomposition is a function of its argument alone, so Z_{t+1} agrees. The state carried to the next step, namely M_t, V_t^{row} and V_t^{col}, has already been shown to agree. This closes the induction. □

Corollary 1. *Under the same assumptions, the convention used to split Z_{t+1} back into factors does not change $\{Z_t\}$. Writing $B \leftarrow U\Sigma^{1/2}$, $A \leftarrow \Sigma^{1/2}V^{\top}$ and writing $B \leftarrow U\Sigma$, $A \leftarrow V^{\top}$ give the same trajectory.*

Proof. Both conventions produce a factorization of the same matrix $Z_{t+1} = U\Sigma V^{\top}$, and the two are related by an invertible $R' \in \mathbb{R}^{r\times r}$: taking $(B, A) = (U\Sigma^{1/2},\, \Sigma^{1/2}V^{\top})$ and $(B', A') = (U\Sigma,\, V^{\top})$ we have $B' = B\,\Sigma^{1/2}$ and $A' = \Sigma^{-1/2}A$, so $R' = \Sigma^{1/2}$, which is invertible because Z_{t+1} has rank r. Apply the induction above with R' in place of R and with step $t+1$ as the base case: Z_{t+1} is the same on both sides by construction, and the carried state $M_t, V_t^{\mathrm{row}}, V_t^{\mathrm{col}}$ was produced before the write-back and so does not depend on it. Every later quantity is a function of those, so the two trajectories coincide from $t+1$ onward. □

C TEMPER STEP BY STEP

Algorithm 1 states one step of Temper in terms of the equations of Section 3.

Algorithm 1 Temper, one step for one adapted layer; $G = PQ^\top$ is never formed.

Require: factors (B, A); gradients $(\nabla B, \nabla A)$; state $\dot{M}_B, \dot{M}_A, V^{\mathrm{row}}, V^{\mathrm{col}}$ and previous frames $\tilde{U}_B, \tilde{U}_A$; step index t; learning rate η

1: $U_B, R_B \leftarrow \mathrm{polar}(B)$; $\quad U_A, R_A \leftarrow \mathrm{polar}(A^\top)$
2: $\bar{\nabla} B \leftarrow \nabla B\, R_A^\dagger$; $\quad \bar{\nabla} A \leftarrow R_B^\dagger\, \nabla A$ (2)
3: $\dot{B} \leftarrow (I - U_B U_B^\top)\bar{\nabla} B$; $\quad \dot{A} \leftarrow \bar{\nabla} A$; $\quad P \leftarrow [\dot{B}, U_B]$; $\quad Q \leftarrow [U_A, \dot{A}^\top]$ (7)
4: $V^{\mathrm{row}} \leftarrow \beta_2 V^{\mathrm{row}} + (1{-}\beta_2)\,\mathrm{diag}(PQ^\top QP^\top)$; $\quad V^{\mathrm{col}} \leftarrow \beta_2 V^{\mathrm{col}} + (1{-}\beta_2)\,\mathrm{diag}(QP^\top PQ^\top)$ (12)
5: $M_t \leftarrow \beta_1 P_{Z_t}(M_{t-1}) + (1{-}\beta_1) G_t$, on the pair $(\dot{M}_B, \dot{M}_A)$ (16)
6: $F_t \leftarrow P_{Z_t}\big(c_t\, D_t^{\mathrm{row}}\, \hat{M}_t\, D_t^{\mathrm{col}}\big)$, on the pair (14)
7: $U, \Sigma, V \leftarrow \mathrm{SVD}_r(Z_t - \eta F_t)$, on the $2r \times 2r$ core (17)
8: $B \leftarrow U\Sigma^{1/2}$; $\quad A \leftarrow \Sigma^{1/2} V^\top$; $\quad \tilde{U}_B \leftarrow U_B$; $\quad \tilde{U}_A \leftarrow U_A$

D Complexity

D.1 Time

Table 7: Cost of one step for one layer, by line of Algorithm 1.

Line	Stage	Flops
1	Polar decompositions of B and $A^\top$	$4mr^2 + 4nr^2$
2	Unmagnified gradients (2)	$2mr^2 + 2nr^2$
3	Tangent factors P and Q (7)	$4mr^2$
4	Marginals (12)	$16mr^2 + 16nr^2$
5	Transport and first moment (16)	$12mr^2 + 8nr^2 + O((m{+}n)r)$
6	Two-sided scaling and reprojection (14)	$10mr^2 + 8nr^2 + O(m{+}n)$
7	Retraction (17)	$8mr^2 + 12nr^2 + O(r^3)$
8	Write-back	$O((m{+}n)r)$
	Total	$56mr^2 + 50nr^2 + O\big(r^3 + (m{+}n)r\big)$

Three things follow from Table 7. The step is $O\big((m{+}n)r^2 + r^3\big)$, which is LoRA-RITE's order and not RAdamW's $O(mnr)$. No stage touches an $m \times n$ object: every line works on matrices of size $m \times r$, $n \times r$, $r \times r$ or $2r \times 2r$, so the mn transient allocation that forming G would need never appears. And the r^3 terms, which come from the two $r \times r$ eigendecompositions, the two pseudo-inverses and the $2r \times 2r$ singular value decomposition, are below one percent of the total at $r = 16$ and $m, n \geq 1024$.

The share of a training step that the optimizer accounts for is set by the ratio of r to the number of tokens. Let T be the tokens processed per optimizer step, the effective batch times the sequence length. The adapted layer's own backward pass costs about $4T(m{+}n)r$ flops, so by Table 7 the step adds

$$\frac{56mr^2 + 50nr^2}{4T(m{+}n)r} \;\approx\; \frac{14\,r}{T}, \tag{38}$$

independent of m and n. At $r = 16$ and the $T = 16384$ tokens per step of Appendix E.1 this is 1.3% of the adapter's own backward pass, and about 10^{-4} of the frozen layer's forward and backward pass at $m = n = 3072$. The optimizer runs once per step while the forward and backward run over the whole effective batch, so the share falls as $1/T$. Equation (38) is a ratio of flop counts. At these matrix sizes wall-clock is not proportional to it, so what the step costs in practice is what Table 5 measures.

D.2 Memory

Temper carries one first moment as a tangent pair, the previous step's two frames, and the two marginals:

$$\underbrace{(m{+}n)r}_{\dot{M}_B,\,\dot{M}_A} + \underbrace{(m{+}n)r}_{\tilde{U}_B,\,\tilde{U}_A} + \underbrace{m+n}_{V^{\mathrm{row}},\,V^{\mathrm{col}}} = 2(m{+}n)r + (m{+}n). \tag{39}$$

Adam on the LoRA factors keeps a first and a second moment for each of B and A, which is $2(m{+}n)r$, so (39) is Adam's state plus one number per row and one per column,

$$\frac{2(m{+}n)r + (m{+}n)}{2(m{+}n)r} = 1 + \frac{1}{2r} = 1.031 \quad \text{at } r = 16. \tag{40}$$

Table 5 measures 191.3 MiB against 185.5 MiB, a ratio of 1.031. The transient workspace is P, Q and a few matrices of size $m \times r$ or $n \times r$, so $O((m{+}n)r)$, and no $m \times n$ buffer is allocated at any point.

Table 8: Optimizer state and time per step per layer, in the notation of this section. ∇Z is the gradient with respect to the merged update, which Proposition 3 shows the factor gradients do not determine.

Method	**State**	**Time per step**	**Needs ∇Z**
LoRA with AdamW	$2(m{+}n)r$	$O((m{+}n)r)$	no
LoRA-RITE	$2(m{+}n)r + O(r^2)$	$O((m{+}n)r^2)$	no
LORO	$4(m{+}n)r$	$O((m{+}n)r^2)$	no
RAdamW	$mn + (m{+}n)$	$O(mnr)$	yes
Riemannion	$2(m{+}n)r$	$O((m{+}n)r^2)$	no
Temper (ours)	$2(m{+}n)r + (m{+}n)$	$O((m{+}n)r^2)$	no

The last column is the one that separates the two rows with $m{+}n$ scales. Temper and RAdamW both carry one scale per row and one per column, but Temper reads them from G through (12) and RAdamW reads them from ∇Z, which has to be captured by a backward hook on the frozen weight and materialised in full. That is the difference between $O((m{+}n)r^2)$ and $O(mnr)$ in time, and between $m{+}n$ and mn extra numbers of state, and it is what Table 5 measures as 191.3 MiB against 5480 MiB.

Two costs are outside this accounting. The frozen base model dominates the memory of every arm equally and is reported as part of the peak in Table 5. Communication is not modelled: Temper's state is sharded like any other per-layer optimizer state, and its extra $m + n$ numbers per layer are smaller than the factors themselves by $2r$.

E Experimental details

Every value below is read from the saved per-run configuration of each arm rather than from the repository defaults, which several arms override. Where a value differs between arms it is given per arm.

E.1 Training protocol

Table 9: Per-task training configuration. The schedule is constant with no warm-up in every case, the effective batch is 32, and the gradient-norm clip is 1.0 for every optimizer except LoRA-RITE and RAdamW.

Task	**Epochs**	**Max length**	**Max prompt length**
Commonsense	1	512	256
Code	2	1024	256
GSM8K	3	512	256

E.2 Optimizers

Temper. The iterate is the rank-r matrix Z itself, stored as two orthonormal frames and a small core. The projection, the transport by reprojection and the retraction through the $2r \times 2r$ core are those of Bogachev et al. (2026); what is under test is the preconditioner, which replaces that method's orthogonalization by the factored second moment of (14), so that the rank-$2r$ structure survives it and the two marginals cost $O((m+n)r^2)$ through Gram matrices instead of $O(mn)$.

Table 10: Temper's settings.

Quantity	Value	Note
β_1	0.9	on the transported tangent vector, as in (16), with bias correction $1-\beta_1^t$
β_2	0.9	on the two marginals, with bias correction $1-\beta_2^t$
Weight decay	0	
Learning rate	1×10^{-5}	selected by the sweep of Appendix E.3

The second moment is accumulated on the raw tangent vector G_t rather than on the momentum, which is the structure Adam has; the momentum itself is transported and normalized exactly as (16) states.

E.3 Learning-rate selection

Learning rates do not transfer between these optimizers. The Riemannion step fixes its step norm to $\sqrt{2r}$ and needs a large rate, Temper divides by a second moment and needs a small one, and RAdamW's preconditioner is not Adam's $\sqrt{v}$. Each optimizer was therefore swept on its own grid, and published values from the baselines' own papers were not reused: where they were tried they were wrong for this task, with LoRA-RITE's Gemma-2B grid $\{2 \times 10^{-4}, 5 \times 10^{-4}, 10^{-3}\}$ lying entirely above the usable range here and RAdamW's GPT-2 setting of 8×10^{-3} diverging, as does 3×10^{-3}.

Selection ran in two stages. A 100-step probe on 3200 examples, with no evaluation, rejects learning rates by the median training perplexity over its last 20 steps, and the survivors are promoted to full arms scored on all items. The probe is used only to reject. A probe winner sitting at the edge of its grid is refused promotion, since an endpoint winner is only the best of what happened to be tried. Every selected rate is interior to its grid, so each optimizer's optimum is bracketed on both sides. Outside these ranges the runs are not usable: below the smallest rate listed the training loss barely moves within the budget, and above the largest it diverges.

Table 11: Learning rates run per optimizer. Selected learning rates are in bold.

Optimizer	Learning rates
LoRA with AdamW	1×10^{-5}, $\mathbf{1 \times 10^{-4}}$, 2×10^{-4}, 3×10^{-4}
LoRA-RITE	5×10^{-5}, $\mathbf{1 \times 10^{-4}}$, 3×10^{-4}, 5×10^{-4}
LORO	3×10^{-3}, $\mathbf{1 \times 10^{-2}}$, 3×10^{-2}, 1×10^{-1}
RAdamW	3×10^{-5}, 1×10^{-4}, $\mathbf{3 \times 10^{-4}}$, 1×10^{-3}
Riemannion	3×10^{-4}, 1×10^{-3}, $\mathbf{3 \times 10^{-3}}$, 1×10^{-2}
Temper	1×10^{-6}, 3×10^{-6}, $\mathbf{1 \times 10^{-5}}$, 5×10^{-5}

F Limitations

Our evaluation covers rank 16 on two models of comparable size and three task families, in the fine-tuning setting. Other ranks, other model scales, pre-training, and the combination with methods that reallocate rank across layers are natural next steps that we leave to future work. Temper also operates at a fixed rank, since it steps on the manifold of rank-r matrices; letting the rank adapt during training is an interesting extension. Our analysis establishes invariance and two bounds on the row profile of the update, and convergence on this manifold is beyond its scope.