RLOptimArena: Is Your Optimizer Properly Designed for RLVR?

Anonymous ACL submission

Abstract

What kind of optimizer is best suited for Reinforcement Learning with Verifiable Rewards (RLVR)? Current practice inherits the optimizers of large-scale pretraining *e.g.,* Adam, momentum, second-moment buffers, and matrix preconditioners, yet RLVR is a markedly different regime, with sparse, low-rank, off-principal updates and a comparatively flat loss surface, leaving it unclear whether this machinery is still the right choice. We introduce RLOptimArena, a fair and reproducible benchmark that evaluates seven mainstream optimizers under RLVR. We find that optimizers do not lie on a smooth quality spectrum but fall into four discrete regimes—healthy, weak learning, not learning, and crashed—and that, contrary to the expectation that RLVR's lightweight updates favor lightweight optimizers, the most memory-frugal options fail while full adaptivity (Adam) or a well-conditioned matrix update (Muon) remains essential. To explain this, we propose an energy-allocation framework that splits each optimizer's update into the energy aligned with a shared RL direction and the energy spent off it; the four regimes are exactly the cells of the resulting alignment–magnitude plane.

1 Introduction

Optimizers are a central component of neural network training. As large language models (LLMs) scale has grown to hundreds of billions of parameters (Achiam et al., 2023; Yang et al., 2025; Team et al., 2025; Zeng et al., 2026), the choice of optimizer has become increasingly consequential to the cost, stability, and final quality of model training. Prior work has produced a rich landscape of optimizers. SGD and its momentum variants (Robbins and Monro, 1951) are simple and generalize well, but struggle with the heterogeneous gradient scales in deep networks. Adaptive methods such as Adam and AdamW (Kingma and Ba, 2014) address this via per-parameter learning rates and have become the de facto standard. Lion (Chen et al., 2023) achieves competitive results with far less memory using sign-based updates and a single momentum buffer, while Shampoo (Gupta et al., 2018) and related methods exploit second-order curvature via Kronecker-factored statistics. Most recently, Muon (Liu et al., 2025; Jordan et al., 2024) applies Newton-Schulz orthogonalization to momentum matrices, yielding well-conditioned updates that perform strongly in frontier-model pretraining.

In parallel, Reinforcement Learning with Verifiable Rewards (RLVR) (Guo et al., 2025; Shao et al., 2024) has emerged as a critical pathway for advancing LLM capabilities. By optimizing against verifiable signals with algorithms such as GRPO (Guo et al., 2025), RLVR unlocks test-time scaling on reasoning-intensive tasks, driving the dramatic gains of recent reasoning models. Crucially, mounting evidence indicates that RLVR differs fundamentally from the teacher-forcing paradigm of SFT and pretraining. Mukherjee et al. (2026a); Balashov (2025) show that RLVR finetunes only a strikingly small subnetwork, leaving most parameters untouched. Zhu et al. (2025); Yin et al. (2025) demonstrate that RLVR provably learns *off* the principal directions in weight space—achieving its gains via minimal spectral drift and off-principal update alignment—whereas SFT targets principal weights, distorts the spectrum, and even underperforms RLVR. Recent works also report that the acquired capability is largely low-rank (Wang et al., 2025), capturable by a single transferable reasoning vector (Zbeeb et al., 2025) that can even be linearly extrapolated for stronger performance. Complementing these structural findings, Mukherjee et al. (2026b) argue that the RLVR loss landscape is markedly flatter than pretraining's, and show that simpler optimizers like SGD can match or surpass Adam here at a fraction of the memory and compute cost.

These observations have a practical implication

Optimizer	1st-order state	2nd-order / preconditioner	Parameter update
SGD	—	—	$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \eta\, \mathbf{g}_t$
RMSProp	—	$\mathbf{v}_t = \beta\, \mathbf{v}_{t-1} + (1-\beta)\, \mathbf{g}_t^2$	$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \eta\, \dfrac{\mathbf{g}_t}{\sqrt{\mathbf{v}_t} + \varepsilon}$
Adam	$\mathbf{m}_t = \beta_1 \mathbf{m}_{t-1} + (1-\beta_1)\mathbf{g}_t$ $\hat{\mathbf{m}}_t = \mathbf{m}_t/(1-\beta_1^t)$	$\mathbf{v}_t = \beta_2 \mathbf{v}_{t-1} + (1-\beta_2)\mathbf{g}_t^2$ $\hat{\mathbf{v}}_t = \mathbf{v}_t/(1-\beta_2^t)$	$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \eta\, \dfrac{\hat{\mathbf{m}}_t}{\sqrt{\hat{\mathbf{v}}_t} + \varepsilon}$
Lion	$\mathbf{c}_t = \beta_1 \mathbf{m}_{t-1} + (1-\beta_1)\mathbf{g}_t$ $\mathbf{m}_t = \beta_2 \mathbf{m}_{t-1} + (1-\beta_2)\mathbf{g}_t$	—	$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \eta\big(\mathrm{sign}(\mathbf{c}_t) + \lambda\, \boldsymbol{\theta}_{t-1}\big)$
Muon	$\mathbf{m}_t = \mu\, \mathbf{m}_{t-1} + \mathbf{g}_t$	$\mathbf{O}_t = \mathrm{NewtonSchulz}(\mathbf{m}_t)$ $\approx (\mathbf{m}_t \mathbf{m}_t^\top)^{-1/2} \mathbf{m}_t$	$\mathbf{W}_t = \mathbf{W}_{t-1} - \eta\, \kappa\, \mathbf{O}_t$
Shampoo	—	$\mathbf{L}_t = \mathbf{L}_{t-1} + \mathbf{G}_t \mathbf{G}_t^\top$ $\mathbf{R}_t = \mathbf{R}_{t-1} + \mathbf{G}_t^\top \mathbf{G}_t$	$\mathbf{W}_t = \mathbf{W}_{t-1} - \eta\, \mathbf{L}_t^{-1/4} \mathbf{G}_t \mathbf{R}_t^{-1/4}$
SOAP	$\mathbf{M}'_t = \beta_1 \mathbf{M}'_{t-1} + (1-\beta_1)\mathbf{G}'_t$	$\mathbf{L}_t = \beta_2 \mathbf{L}_{t-1} + (1-\beta_2)\mathbf{G}_t \mathbf{G}_t^\top$ $\mathbf{R}_t = \beta_2 \mathbf{R}_{t-1} + (1-\beta_2)\mathbf{G}_t^\top \mathbf{G}_t$ $\mathbf{V}'_t = \beta_2 \mathbf{V}'_{t-1} + (1-\beta_2)\mathbf{G}_t'^2$	$\mathbf{G}'_t = \mathbf{Q}_L^\top \mathbf{G}_t\, \mathbf{Q}_R$ $\mathbf{W}_t = \mathbf{W}_{t-1} - \eta\, \mathbf{Q}_L\, \widetilde{\mathbf{M}}_t\, \mathbf{Q}_R^\top$ where $\widetilde{\mathbf{M}}_t = \dfrac{\mathbf{M}'_t}{\sqrt{\mathbf{V}'_t} + \varepsilon}$

Table 1: Per-step update rules of common optimizers. Element-wise optimizers act on parameters $\boldsymbol{\theta}$ with vector gradient $\mathbf{g}_t$; matrix-aware optimizers act on weight matrices $\mathbf{W}$ with matrix-shaped gradient $\mathbf{G}_t$. Column colors mark first-order state, second-order/preconditioner, and parameter update; the update column is further split into element-wise ($\boldsymbol{\theta}$) and matrix ($\mathbf{W}$) updates. η: learning rate; ε: numerical constant; $\mu, \beta, \beta_1, \beta_2$: decay/momentum coefficients. —denotes an absent term. κ: a shape-dependent scale factor matching Muon's update RMS to Adam's; λ: weight-decay coefficient; Q_L, Q_R: the orthogonal eigenbasis matrices of the left/right preconditioners L_t, R_t, in which SOAP rotates the gradient before applying the Adam update.

for optimizer design. The optimizers with *e.g.,* large momentum, second-moment buffers, preconditioning *etc.,* used in modern LLM training were developed for pretraining where the loss landscape is rugged and the data is noisy and heterogeneous. RLVR looks different: updates are sparse, low-rank, off-principal, and the loss surface is comparatively flat. It is therefore unclear whether the standard optimizer machinery is still the right choice, which leads to the question we study:

> *What kind of optimizer is best suited for RLVR?*

To answer this question, we introduce **RLOptimArena**, a fair and reproducible benchmark for evaluating mainstream optimizers in the RLVR setting.Across seven optimizers on math and code, RLVR optimizer behavior does not form a smooth quality spectrum but falls into four discrete regimes: *healthy* (Adam, Muon), *weak learning* (SOAP), *not learning* (SGD, RMSProp), and *crashed* (Lion). Contrary to the expectation that RLVR's lightweight updates favor lightweight optimizers, the most memory-frugal options fail, while full adaptivity (Adam) or a well-conditioned matrix update (Muon) remains essential. To explain this, we propose an *energy-allocation* framework that decomposes each optimizer's update into the energy aligned with a shared RL direction and the energy spent off it, and show that the four regimes are exactly the cells of the resulting alignment–magnitude (γ, E) plane. The rest of the paper is organized as follows.

- Section 2 describes the RLVR algorithm, the optimizers under comparison, and the unified training and evaluation protocol.
- Section 3.1 reports the four-regime result and shows that maintaining a first-order moment is the empirical divide between learning and not learning.
- Section 3.2 introduces the energy-allocation framework, and Section 3.3 uses it to account for all four regimes.

2 Preliminaries & Setup

In this section we describe the experimental setup: the RLVR algorithm we optimize against, the optimizers under comparison, the training and evaluation data, and the hyper-parameter protocol that ensures a fair comparison across optimizers.

RLVR. Reinforcement Learning with Verifiable Rewards (RLVR) fine-tunes a language-model policy against rewards that can be checked programmatically. As our base algorithm we adopt a variant that resembles GRPO (Shao et al., 2024) but

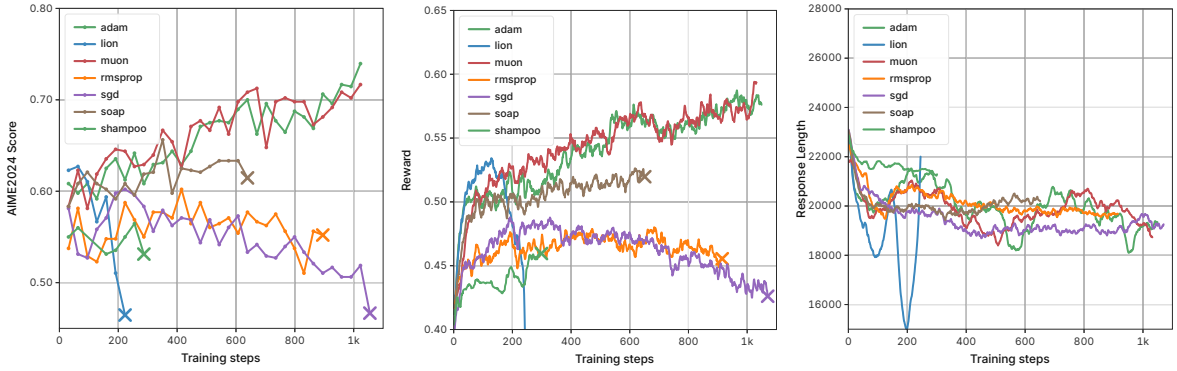

Figure 1: Training dynamics on math RLVR for the seven optimizers of Table 1. *Left:* downstream AIME2024 accuracy across training steps. *Center:* rollout reward on the math training set. *Right:* mean response length. A cross ($\times$) marks a training collapse, i.e. an abrupt spike in gradient norm. Rollout reward and downstream accuracy move together throughout, and the four regimes are already apparent: Adam and Muon improve steadily, SOAP gains little, SGD and RMSProp stall, and Lion collapses early.

	AIME2024	AIME2025	AIME2026	Beyond AIME	HMMT Feb 2026	HMMT Nov 2025	Avg.
SGD	48.2	36.9	42.4	20.5	26.3	30.4	34.1
Adam	69.2	60.1	64.7	35.0	38.6	48.3	52.6
RMSProp	54.8	42.1	50.1	21.9	29.2	36.2	39.1
Shampoo	53.0	47.0	54.0	26.0	31.0	38.0	41.5
Lion	47.6	37.6	42.3	20.3	25.2	30.9	34.0
SOAP	61.1	49.6	58.6	30.6	34.9	40.5	45.9
Muon	69.2	57.6	64.8	35.0	39.8	47.2	52.3

Table 2: Accuracy (%) on mathematical-reasoning benchmarks after RLVR. **Avg.** is the mean over the six benchmarks. For optimizers that destabilize or decline before the 1,000-step budget, we report the best checkpoint prior to collapse.

omits the KL-regularization term, in line with recent large-scale training reports (Team et al., 2025; Blakeman et al., 2025; Zeng et al., 2026). Following current best practice, we additionally adopt the asymmetric clipping of DAPO (Yu et al., 2026), which has become a default mechanism for preventing entropy collapse and preserving output diversity.

Concretely, for each prompt x, the old policy $\pi_{\text{gen}}(\theta_{\text{old}})$ samples G candidate completions $\{y_i\}_{i=1}^{G}$, each with a scalar verifiable reward r_i. We compute group-normalized advantages by centering and standardizing rewards within each group:

$$\hat{A}_i^G = \big(r_i - \text{mean}(\{r_j\}_{j=1}^G)\big) / \big(\text{std}(\{r_j\}_{j=1}^G) + \epsilon\big). \quad (1)$$

RLVR maximizes the expected verifiable reward $\mathbb{E}_{x\sim D,\, y\sim\pi_\theta}[\, r(x,y)\,]$. We optimize a clipped policy-gradient surrogate, applying asymmetric thresholds ϵ^-, ϵ^+ to the token-level importance-sampling ratio $\rho_{i,t}$:

$$\rho_{i,t} := \frac{\pi^{\theta}_{\text{train}}(y_{i,t} \mid x, y_{i,<t})}{\pi^{\theta_{\text{old}}}_{\text{gen}}(y_{i,t} \mid x, y_{i,<t})}, \quad (2)$$

$$\text{clip}_{\text{a}}(\rho) := \text{clip}(\rho,\, 1-\epsilon^-,\, 1+\epsilon^+). \quad (3)$$

Aggregating per-token losses at the *sample level* yields the surrogate objective. The surrogate is then

$$\mathcal{J}(\theta) = \mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|y_i|}\sum_{t=1}^{|y_i|} L_{i,t}(\theta)\right], \quad (4)$$

$$L_{i,t}(\theta) = \min\big(\rho_{i,t}\hat{A}_i^G,\, \text{clip}_{\text{a}}(\rho_{i,t})\hat{A}_i^G\big). \quad (5)$$

We utilize the on-policy strategy (Shao et al., 2024) for our RLVR training which is different from async settings (Zeng et al., 2026).

Optimizers. We benchmark a representative set of optimizers spanning the major families used in modern LLM training; their update rules are summarized in Table 1. *Plain SGD* (Robbins and Monro, 1951) applies the raw stochastic gradient with no auxiliary state—the simplest and most memory-frugal option, but slow to converge under the heterogeneous gradient scales of deep networks. *Adaptive methods*, most prominently RMSProp (Graves, 2013) and Adam/AdamW (Kingma and Ba, 2014), address this with per-parameter second-moment estimates and have become the default for LLM pretraining, at the cost of substantial optimizer-state memory. *Sign-based meth-*

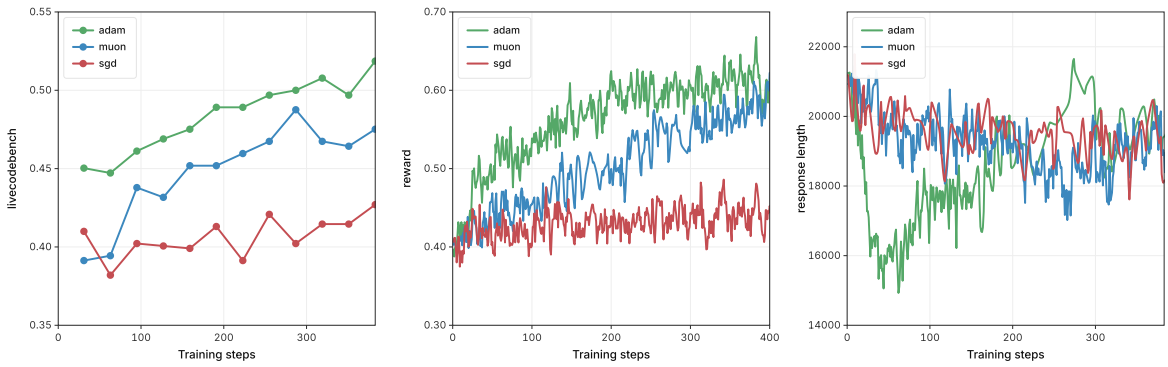

Figure 2: Training dynamics on code RLVR for the three optimizers of Table 1. *Left:* downstream LiveCodeBench accuracy across training steps. *Center:* rollout reward on the code training set. *Right:* mean response length. A cross ($\times$) marks a training collapse, *i.e.,* an abrupt spike in gradient norm.

	LCB v4	LCB v5	LCB v6	avg
SGD	30.9	30.9	35	32.2
Adam	36.3	43.2	41.4	40.3
Muon	36.1	40.2	39.1	38.4

Table 3: Accuracy (%) on code-generation benchmarks after RLVR.

ods such as Lion (Chen et al., 2023) cut this overhead by retaining only a single momentum buffer and updating along the sign of the gradient. *Matrix-preconditioned methods* instead exploit gradient geometry: Shampoo (Gupta et al., 2018) and SOAP (Vyas et al., 2025) apply Kronecker-factored preconditioners that approximate second-order curvature, while Muon (Jordan et al., 2024; Liu et al., 2025) orthogonalizes momentum matrices via Newton–Schulz iteration and has been shown to scale to billion-parameter pretraining at roughly half the compute of AdamW.

Data and Benchmarks. We center our study on the two scenarios most widely used to probe LLM reasoning: mathematics and code. For *Cold-start SFT*, the supervised cold-start stage uses `allenai/Dolci-Think-SFT-7B` (Olmo et al., 2025), a fully open-source reasoning-mode dataset spanning STEM, code, and everyday dialogue; its broad coverage keeps the starting checkpoint close to a realistic training distribution rather than narrowly specializing it. For *RL training data*, for mathematics we train on DAPO-Math-17K (Yu et al., 2026); for code we use the coding subset of the Nemotron-3 RL data (Blakeman et al., 2025). For *Evaluation*, we evaluate mathematical reasoning on AIME2024, 2025, and 2026, Beyond AIME, and HMMT, and code generation on LiveCodeBench (Jain et al., 2025). We evaluate these benchmarks with average@32 *i.e.,* we average the accuracy of 32 runs for each query, for math benchmarks and average@4 for coding benchmarks. This ensures stable results and reproductivity of our experiments. We deliberately exclude agentic coding benchmarks such as SWE-bench Verified and SWE-bench Pro: reliable evaluation there depends heavily on the surrounding harness and on long agentic rollouts, which makes outcomes difficult to attribute cleanly to the optimizer and is prohibitively expensive to run across our full optimizer sweep.

Model. We utilize the base model version of Qwen3 8B (Yang et al., 2025) as the backbone for all experiments. To ensure full transparency and reproducibility, we cold-start the base model with a SFT stage on `Dolci-Think-SFT-7B` (Olmo et al., 2025). This stage equips the model with three properties required for stable RLVR: coherent step-by-step reasoning, basic conversational ability, and consistent emission of the thinking tags that delimit its reasoning trace. Crucially, the same cold-start checkpoint is used to initialize every optimizer run, so that all RLVR comparisons start from an identical capability profile.

Hyperparameters and Training Details. We tune the learning rate per optimizer, starting from $1e{-}6$ and searching upward and downward to locate the best value for each; following common practice, we use no warmup. All runs share the remaining hyper-parameters: a weight decay of 0.1, a maximum reasoning length of 30K tokens, and a batch of 64 independently sampled prompts per step with 8 rollouts each. Every optimizer is trained for a fixed budget of $1{,}000$ steps to ensure a controlled comparison. To accelerate training, we additionally adopt a data-profiling strategy: before

RL training, we roll out the cold-start SFT model on the training set and discard prompts it already solves with accuracy 1, since such prompts are removed by DAPO's dynamic sampling filter (Yu et al., 2026) and would otherwise stall throughput.

3 Experiments and Analysis

We evaluate the seven optimizers of Table 1 under the unified protocol of Section 2. We first report what happens (Section 3.1), then build an *energy-allocation* account of why (Sections 3.2–3.3).

3.1 Experimental Results

Math. The seven optimizers separate cleanly into four regimes. Only Adam and Muon are healthy: both improve near-monotonically and stay stable across the full 1,000-step budget. Adam is strongest, with average downstream accuracy reaching 52.6% at a steady reward slope of $+0.087$ per 1,000 steps; Muon tracks it closely, finishing at 52.3%. SOAP and Shampoo are weak learners: both improve early but their gains stall, peaking at 45.9% and 41.5% respectively. SGD and RMSProp never learn—RMSProp, adaptive but momentum-free, stalls on a $\sim$39% plateau, while SGD gains briefly before declining to 34.1% (-11 points per 1,000 steps). Lion crashes: its sign-based update destabilizes almost immediately and collapses to 34.0%, the lowest of the seven, despite the highest starting accuracy. Across all optimizers, rollout reward and downstream accuracy move together (Figure 1), indicating the reward faithfully reflects optimization progress rather than evaluation noise or reward hacking.

Code. The code setting reproduces the math picture on the three optimizers we sweep (SGD, Adam, Muon; Table 3). Adam and Muon are again healthy, with reward and LiveCodeBench accuracy rising together; Adam finishes strongest (40.3% average over LCB v4–v6), Muon close behind (38.4%). SGD again fails to learn, stalling at 32.2%—8 points below Adam—just as it does on math. The healthy/not-learning split thus holds across domains, supporting Section 3.1's claim that the four regimes are a property of the optimizer under RLVR rather than of the task.

Learning-rate robustness. One might worry that the four regimes are simply caused by learning-rate mis-tuning *e.g.,* that SGD or Lion would become healthy under a better η. To test this, we

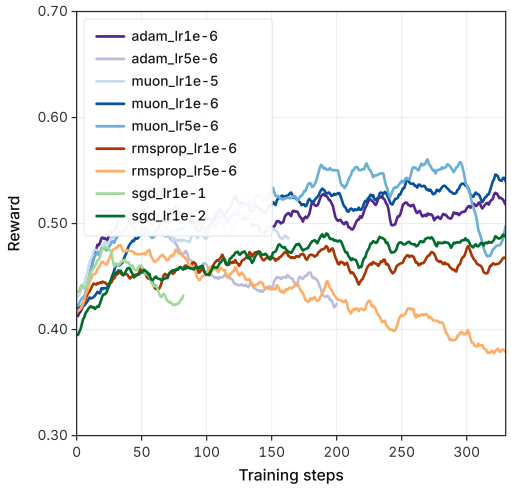

Figure 3: Learning-rate robustness.

sweep η around each optimizer's best value and compare the reward (Figure 3). The qualitative regime is stable under this sweep. Adam and Muon remain the only healthy optimizers, showing sustained improvements in reward and downstream accuracy. SOAP remains a weak learner: tuning changes its peak but does not make its gains persistent. SGD and RMSProp do not achieve sustained learning, and Lion remains unstable, with learning rate mainly shifting the timing rather than the existence of collapse. Thus, η controls the speed and severity of each run, but not the optimizer's regime assignment.

The four regimes and the role of a first-order moment. The seven optimizers split into four regimes. Adam and Muon are *healthy*: entropy decreases steadily, reward rises, and downstream accuracy stays above 69%, with stable dynamics across the full 1,000-step budget. SOAP is a *weak learner*—it moves in the right direction, but entropy is nearly flat and reward gains only $\sim+0.02$, too slow to surface downstream. SGD and RMSProp *never learn*: entropy fails to decrease and reward stalls or declines. Lion, and RMSProp at larger learning rates, *crash*, a sudden spike in gradient norm and a rebound in entropy. What decides this outcome is whether the optimizer maintains a first-order moment. Every optimizer that accumulates gradients into a momentum buffer (Adam, Lion, SOAP, Muon) settles on a consistent improvement direction, while those acting on the raw gradient or a per-coordinate rescaling of it (SGD, RMSProp) do not. This is NOT consistent with Mukherjee et al. (2026b) which demonstrates the effectiveness of SGD. A first-order moment

is necessary but not sufficient, however, Lion and SOAP keep one yet still crash or stall—so the update must additionally be *scaled and conditioned* well, which is what separates Adam and Muon from the rest.

Comparison with prior reports. Our finding that SGD fails appears to contradict recent reports that momentum- and adaptivity-free SGD suffices for RLVR (Mukherjee et al., 2026b). The discrepancy, however, traces directly to differences in experimental regime rather than to a genuine disagreement. Mukherjee et al. (2026b) start from already instruction-tuned checkpoints, generate short rollouts of roughly 3K tokens, and train for only 200–500 update steps; RLOptimArena instead starts from a cold-start reasoning model, uses reasoning rollouts of up to 30K tokens, and trains for 1,000 steps. In the short-rollout, few-step regime, RLVR makes only small, localized parameter changes, for which the raw stochastic gradient is an adequate update. Long reasoning trajectories change this picture: per-step gradient variance grows with sequence length and with the credit-assignment horizon, and without a first-order moment to average it away, SGD's update direction is progressively dominated by noise—an instability of long-horizon optimization that prior work has documented (Qi et al., 2025; Wang et al., 2026; Zeng et al., 2026). Consistent with this account, we observe SGD improving early and then steadily declining: the lightweight regime is real but narrow, and is not robust to the longer, harder RLVR runs that RLOptimArena is designed to probe.

Finding 1. The seven optimizers do not lie on a smooth quality spectrum but fall into four discrete regimes—*healthy* (Adam, Muon), *weak learning* (SOAP), *not learning* (SGD, RMSProp), and *crashed* (Lion)—and the assignment is stable under learning-rate tuning. Maintaining a first-order moment is what separates learning from not learning: averaging noisy policy gradients over time recovers a consistent update direction. It is necessary but not sufficient—the averaged update must also be well scaled and conditioned—which is why only Adam and Muon remain healthy.

3.2 The Energy-Allocation Framework

The four regimes of Section 3.1 are behavioral labels; explaining them requires a quantitative lens that applies to every optimizer alike. We obtain one by asking a single question: *where does an optimizer spend its update?* Since all runs start from the same cold-start checkpoint, their cumulative RLVR updates are directly comparable as task vectors. We formalize this through three definitions.

Definition 1: Task vector & shared direction

Let θ_0 be the shared cold-start checkpoint and $\theta_{\mathrm{RL}}^{(o)}$ the parameters optimizer o reaches under RLVR. Its *task vector* is $\tau_o = \theta_{\mathrm{RL}}^{(o)} - \theta_0$, with normalized direction $u_o = \tau_o / \|\tau_o\|$. The *shared RL direction* $\phi^\star$ is the unit vector maximizing the total squared projection of these normalized directions,

$$\phi^\star = \arg\max_{\|\phi\|=1} \sum_{o \in \mathcal{O}} \langle u_o, \phi \rangle^2, \tag{6}$$

equivalently the top singular vector of the matrix whose rows are the normalized u_o. All inner products and norms are computed over a fixed set of representative weight matrices $\mathcal{M}$, with $\langle a, b \rangle = \sum_{m \in \mathcal{M}} \langle a^{(m)}, b^{(m)} \rangle_F$ and $\|a\|^2 = \sum_{m \in \mathcal{M}} \|a^{(m)}\|_F^2$.

Two points deserve emphasis. First, normalizing each τ_o before the maximization is what makes $\phi^\star$ a *consensus* direction rather than a magnitude-dominated one: a single large-norm failed run cannot pull $\phi^\star$ toward itself, and an optimizer whose updates are dominated by noise shares no consistent direction with the others and contributes little to the objective. $\phi^\star$ is therefore fixed by whichever optimizers agree. A leave-one-out estimate that excludes the optimizer under measurement yields the same direction, ruling out circularity. Second, $\phi^\star$ is computed *across optimizers'* task vectors and must not be conflated with a principal direction of the weight matrices themselves. The two are distinct objects: since RLVR updates are known to concentrate *off* the principal weight directions (Zhu et al., 2025), $\phi^\star$ lies within that off-principal subspace rather than along a principal weight direction.

Definition 2: Energy decomposition

The *update energy* of a task vector τ is $E = \|\tau\|^2$. Writing $\alpha = \langle \tau, \phi^\star \rangle$, it splits into two orthogonal components,

$$E_\phi = \alpha^2, \; E_\perp = \|\tau - \alpha\phi^\star\|^2, \; E = E_\phi + E_\perp, \tag{7}$$

the *aligned energy* spent along $\phi^\star$ and the *residual energy* spent off it.

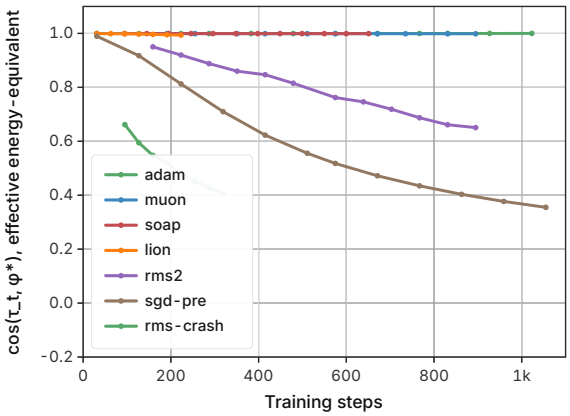

Figure 4: **Direction retention during RLVR.** We plot $c_{o,t} = \cos(\tau_{o,t}, \phi^\star)$ with $\tau_{o,t} = \theta_{o,t} - \theta_0$. Adam, Muon, SOAP, and Lion align with the shared RLVR direction almost immediately and remain near 1.0 throughout training. RMSProp gradually drifts away, while SGD initially aligns but fails to retain the direction; the high-learning-rate RMSProp variant loses alignment fastest. Random directions are nearly orthogonal to $\phi^\star$. Legend labels `rms2`, `sgd-pre`, and `rms-crash` denote RMSProp, SGD, and the high-learning-rate RMSProp variant, respectively.

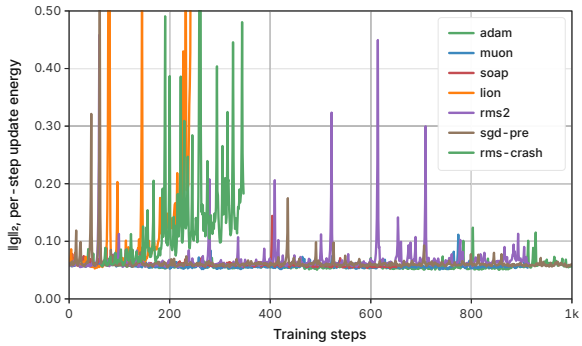

Figure 5: Per-step gradient norm as an instability proxy. Lion collapses through a sharp step-level spike despite a healthy cumulative direction; the high-learning-rate RMSProp variant shows both off-axis energy accumulation and unstable dynamics. See Figure 4 for legend labels.

Our central claim is that the four regimes are precisely the cells of the (γ, E) plane. Alignment comes first: an optimizer with low γ leaks its energy off-axis and cannot learn, whatever its E. Among optimizers with high γ, magnitude decides the outcome—too little energy starves learning, too much destabilizes it, and only an intermediate band is healthy. These two axes are exactly the two clauses of Finding 1: a first-order moment is what raises γ, since averaging cancels the off-axis noise, and good scaling and conditioning are what keep E within the healthy band.

Definition 3: Alignment and magnitude

The *alignment* of an optimizer is $\gamma = E_\phi / E \in [0, 1]$, equivalently $\cos^2(\tau, \phi^\star)$; its *magnitude* is the total update energy E.

3.3 Why the Regimes Arise

We now place every optimizer on the two axes of Section 3.2: first γ (direction), then E (magnitude and step stability).

Direction retention. Figure 4 tracks the per-checkpoint alignment $c_{o,t} = \cos(\tau_{o,t}, \phi^\star)$ along training, where $\tau_{o,t} = \theta_{o,t} - \theta_0$. Direction retention already separates learning from not-learning. Adam, Muon, SOAP, and Lion all lock onto $\phi^\star$ from the first saved checkpoints and remain aligned throughout training. By contrast, RMSProp and SGD progressively lose alignment: RMSProp starts partially aligned but drifts away, while SGD initially lies close to the shared direction but steadily rotates away from it; the high-learning-rate RMSProp variant loses alignment fastest. The not-learning runs are thus not simply making smaller updates—they accumulate updates in directions increasingly different from $\phi^\star$. Their failure is a direction-allocation failure: update energy leaks off-axis instead of concentrating on the shared direction.

Magnitude and step stability. Table 4 decomposes each optimizer's final task-vector energy into E_ϕ and $E_\perp$, and Figure 5 pairs this with the per-step gradient norm as an instability proxy. Adam and Muon spend almost all of their update energy along $\phi^\star$, with negligible off-axis residual and a well-sized E. SOAP also allocates nearly all of its cumulative energy to $\phi^\star$—it is *not* a wrong-direction optimizer; its aligned movement simply does not translate into comparable policy improvement. SGD and RMSProp behave oppositely: their total energy is large but most of it lies in $E_\perp$, so they move substantially in weight space while little of that movement reaches $\phi^\star$. Lion exposes a third pattern: its cumulative task vector stays well aligned and shows no large off-axis leakage, yet the right panel of Figure 5 reveals a sharp per-step gradient-norm spike at the collapse point. Its sign-based update preserves cumulative direction while applying overly aggressive per-step changes under on-policy distribution shift. The four regimes are now exactly the cells of the (γ, E) plane.

Optimizer	E_ϕ	$E_\perp$	E	γ	If Learn
Adam	138.8	≈ 0	138.8	1.000	Healthy
Muon	138.5	0.4	138.9	0.997	Healthy
SOAP	138.5	0.2	138.7	0.998	Weak
Lion	138.5	1.8	140.3	0.987	Crashed
SGD	244.6	1696.2	1940.8	0.126	No
RMSProp	165.5	225.4	390.9	0.423	No
RMSProp†	201.3	1018.4	1219.7	0.165	Crashed

Table 4: **Energy allocation of each optimizer's task vector τ.** $E_\phi = \alpha^2$: energy along $\phi^\star$; $E_\perp = \|\tau - \alpha\phi^\star\|^2$: off-axis residual; $E = E_\phi + E_\perp$: total magnitude; $\gamma = E_\phi/E$: fraction on the shared direction. High γ with mid-range E is healthy; high γ with too-small or too-large E gives weak learning or a crash; low γ never learns. †: high-learning-rate diagnostic variant.

Not learning: energy leaks off-axis (SGD, RMSProp). SGD and RMSProp have low γ: lacking a first-order moment, their per-step noise never cancels, so τ points in no consistent direction and most update energy lands in $E_\perp$. The failure is not too little movement, but off-axis accumulation.

Healthy: concentrated, well-sized energy (Adam, Muon). Adam and Muon combine high γ with an E in the healthy band: a first-order moment locks τ onto $\phi^\star$, and per-parameter adaptivity (Adam) or an orthogonalized update (Muon) keeps the step well-conditioned. Muon allocates its energy most tightly, consistent with being the most stable run.

Weak learning: right direction, too little energy (SOAP). SOAP has high γ but too small an E to move reward appreciably. Alignment alone would label it healthy; the magnitude axis exposes the failure, making SOAP a clean case that correct direction is necessary but not sufficient.

Crashed: right direction, unstable magnitude (Lion). Lion also aligns well but fails on magnitude: its sign-based update applies a uniform, unscaled step, so the energy is aimed correctly yet overshoots and the gradient norm spikes. Lion fails not on direction but on step stability.

Finding 2. The four regimes are the cells of a single (γ, E) plane: γ is the fraction of update energy aligned with the shared RL direction $\phi^\star$, and E its magnitude. The two axes are independent: low γ fails outright (SGD, RMSProp) regardless of E, and among well-aligned optimizers only an intermediate E is healthy (Adam, Muon)— too little starves learning (SOAP), an unscaled step destabilizes it (Lion). Alignment is necessary but not sufficient; the update must also be stably scaled and conditioned.

4 Related Work

The optimizers we benchmark span the major families of LLM pretraining: plain SGD (Robbins and Monro, 1951), adaptive methods (Graves, 2013; Kingma and Ba, 2014), the memory-frugal sign-based Lion (Chen et al., 2023), and matrix-aware preconditioners (Gupta et al., 2018; Vyas et al., 2025; Jordan et al., 2024; Liu et al., 2025)—all developed for pretraining, not RLVR. RLVR, optimized with GRPO (Shao et al., 2024; Guo et al., 2025) and DAPO (Yu et al., 2026), is a distinct regime: it finetunes a small subnetwork (Mukherjee et al., 2026a; Balashov, 2025), learns off the principal weight directions (Zhu et al., 2025; Yin et al., 2025), and acquires low-rank capability (Wang et al., 2025; Zbeeb et al., 2025). Closest to us, Mukherjee et al. (2026b) argue SGD can match Adam under RLVR's flatter landscape; we revisit this with a controlled multi-optimizer benchmark, find it regime-dependent, and give a four-regime account of when an optimizer succeeds.

5 Conclusion

We introduced RLOptimArena, a fair and reproducible benchmark for optimizers under RLVR. Across seven optimizers and both math and code tasks, optimizer behavior falls into four discrete regimes—healthy, weak learning, not learning, and crashed—rather than a smooth quality spectrum. An energy-allocation analysis explains them as the cells of a (γ, E) plane, where γ is the fraction of update energy aligned with a shared RL direction and E its magnitude. Contrary to the expectation that RLVR's lightweight updates favor lightweight optimizers, full adaptivity (Adam) or a well-conditioned matrix update (Muon) remains essential, while the most memory-frugal options fail.

Limitations

While this paper aims to provide a fair and reproducible comparison of optimizers under RLVR, our study has several limitations. First, for each optimizer we tune only the learning rate and adopt standard values for the remaining hyper-parameters; we do not conduct a fine-grained ablation over optimizer-specific parameters such as the momentum and second-moment decay coefficients (β_1, β_2). It is possible that some optimizers benefit from RLVR-specific settings of these parameters, and our rankings should be read with this caveat

in mind. Second, although RLOptimArena covers dense backbones and Adam-pretrained checkpoints, our experiments span a limited range of model architectures and parameter scales (8B). Whether our conclusions transfer to substantially larger models or to other architectural families remains an open question. Third, we restrict our evaluation to mathematical reasoning and code generation with relatively short rollouts, and deliberately exclude long-horizon agentic tasks. The optimization dynamics of long agentic rollouts—with sparser rewards and longer credit-assignment paths—may differ from the settings studied here, and we leave a systematic treatment of optimizer behavior in this regime to future work.

Acknowledgments

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Andrii Balashov. 2025. Reinforcement learning finetunes a sparse subnetwork in large language models. *arXiv preprint arXiv:2507.17107*.

Aaron Blakeman, Aaron Grattafiori, Aarti Basant, Abhibha Gupta, Abhinav Khattar, Adi Renduchintala, Aditya Vavre, Akanksha Shukla, Akhiad Bercovich, Aleksander Ficek, and 1 others. 2025. Nvidia nemotron 3: Efficient and open intelligence. *arXiv preprint arXiv:2512.20856*.

Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, and 1 others. 2023. Symbolic discovery of optimization algorithms. *Advances in neural information processing systems*, 36:49205–49233.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Vineet Gupta, Tomer Koren, and Yoram Singer. 2018. Shampoo: Preconditioned stochastic tensor optimization. In *International Conference on Machine Learning*, pages 1842–1850. PMLR.

Naman Jain, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. 2025. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *International Conference on Learning Representations*, volume 2025, pages 58791–58831.

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

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Jingyuan Liu, Jianlin Su, Xingcheng Yao, Zhejun Jiang, Guokun Lai, Yulun Du, Yidao Qin, Weixin Xu, Enzhe Lu, Junjie Yan, and 1 others. 2025. Muon is scalable for llm training. *arXiv preprint arXiv:2502.16982*.

Sagnik Mukherjee, Lifan Yuan, Dilek Hakkani-Tur, and Hao Peng. 2026a. Reinforcement learning finetunes small subnetworks in large language models. *Advances in Neural Information Processing Systems*, 38:132119–132138.

Sagnik Mukherjee, Lifan Yuan, Pavan Jayasinha, Dilek Hakkani-Tür, and Hao Peng. 2026b. Do we need adam? surprisingly strong and sparse reinforcement learning with sgd in llms. *arXiv preprint arXiv:2602.07729*.

Team Olmo, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, and 1 others. 2025. Olmo 3. *arXiv preprint arXiv:2512.13961*.

Penghui Qi, Zichen Liu, Xiangxin Zhou, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. 2025. Defeating the training-inference mismatch via fp16. *arXiv preprint arXiv:2510.26788*.

Herbert Robbins and Sutton Monro. 1951. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Kimi Team, Yifan Bai, Yiping Bao, Y Charles, Cheng Chen, Guanduo Chen, Haiting Chen, Huarong Chen, Jiahao Chen, Ningxin Chen, and 1 others. 2025. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*.

Nikhil Vyas, Depen Morwani, Rosie Zhao, Itai Shapira, David Brandfonbrener, Lucas Janson, and Sham Kakade. 2025. Soap: Improving and stabilizing shampoo using adam for language modeling. In *International Conference on Learning Representations*, volume 2025, pages 93423–93444.

Shangshang Wang, Julian Asilis, Ömer Faruk Akgül, Enes Burak Bilgin, Ollie Liu, and Willie Neiswanger. 2025. Tina: Tiny reasoning models via lora. *arXiv preprint arXiv:2504.15777*.

Zihan Wang, Chi Gui, Xing Jin, Qineng Wang, Licheng Liu, Kangrui Wang, Shiqi Chen, Linjie Li, Zhengyuan Yang, Pingyue Zhang, and 1 others. 2026. Ragen-2: Reasoning collapse in agentic rl. *arXiv preprint arXiv:2604.06268*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Qingyu Yin, Yulun Wu, Zhennan Shen, Sunbowen Li, Zhilin Wang, Yanshu Li, Chak Tou Leong, Jiale Kang, and Jinjin Gu. 2025. Evaluating parameter efficient methods for rlvr. *arXiv preprint arXiv:2512.23165*.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, and 1 others. 2026. Dapo: An open-source llm reinforcement learning system at scale. *Advances in Neural Information Processing Systems*, 38:113222–113244.

Mohammad Zbeeb, Hasan Abed Al Kader Hammoud, and Bernard Ghanem. 2025. Reasoning vectors: Transferring chain-of-thought capabilities via task arithmetic. *arXiv preprint arXiv:2509.01363*.

Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, and 1 others. 2026. Glm-5: from vibe coding to agentic engineering. *arXiv preprint arXiv:2602.15763*.

Hanqing Zhu, Zhenyu Zhang, Hanxian Huang, DiJia Su, Zechun Liu, Jiawei Zhao, Igor Fedorov, Hamed Pirsiavash, Zhizhou Sha, Jinwon Lee, and 1 others. 2025. The path not taken: Rlvr provably learns off the principals. *arXiv preprint arXiv:2511.08567*.