SPECTRAL-DRIVEN ADAPTIVE-RANK OPTIMIZER FOR LARGE LANGUAGE MODEL REINFORCEMENT LEARNING WITH VERIFIABLE REWARDS

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning with verifiable rewards (RLVR) is reshaping how large language models (LLMs) acquire reasoning capabilities, yet the optimizers used remain largely inherited from teacher-forcing training. This raises a fundamental question: does RLVR induce a distinct optimization geometry that calls for a different optimizer? We investigate this question through the spectral structure of LLM gradients and find that gradient energy progressively concentrates into a shrinking effective subspace, which is characteristic of RLVR, as it is absent under supervised fine-tuning (SFT) and is largely removed when the verifiable reward is replaced by a random one. Moreover, the rank required to capture a fixed fraction of gradient energy steadily decreases over training, revealing that the effective dimensionality of useful updates itself evolves with training. These findings motivate a new optimization principle: rather than prescribing the update rank externally, an optimizer should adapt its rank to the evolving spectral structure of the gradient. We introduce **Ares** (**A**daptive **R**ank-**E**ffective **S**pectral Optimization), which estimates the effective update rank from the gradient's spectral energy distribution, tracks its evolution, and confines the gradient signal driving each update to the resulting low-rank subspace. Across LLMs of different scales, Ares improves accuracy over widely used optimizers and generalizes across RLVR algorithms and tasks. Results suggest that effective optimization may require optimizers to be designed around the geometry and dynamics of the training process.

1 INTRODUCTION

Large language models (LLMs) are increasingly optimized via reinforcement learning with verifiable rewards (RLVR), a central component of training strong reasoning models (DeepSeek-AI, 2025; OpenAI, 2024). Yet the optimizers driving this stage, from AdamW (Loshchilov & Hutter, 2019) to preconditioning and orthogonalization methods (Gupta et al., 2018; Jordan et al., 2024), are largely inherited from supervised training and carried into RLVR essentially unchanged. Rather than assuming that RLVR inherits the optimization geometry of supervised training, we uncover its distinctive gradient geometry and identify the key property that should guide optimizer design.

Why might these updates differ? Unlike teacher forcing, whose signal is anchored to reference continuations, RLVR aggregates signed contributions from sampled trajectories: positive-advantage samples raise their likelihood while negative-advantage ones lower it (Shao et al., 2024). Because this signal is drawn from the policy's own evolving distribution, the update direction depends on sampling and shifts during training, and it typically acts on an already capable policy rather than one still forming its representations. These differences raise an empirical question: does the distinct update dynamics of RLVR leave an observable imprint on the spectral structure of its gradients?

They do. Inspired by previous work Chen et al. (2026), we measure the spectral structure of LLM gradient matrices throughout RLVR with the energy captured by the leading singular components, which describes how concentrated the spectrum is, not the algebraic rank, which is almost always full for a dense gradient. Across RLVR, gradient energy progressively concentrates in the leading singular components (Fig. 2). In our experiments, this behavior is characteristic of RLVR rather than of fine-tuning in general: supervised fine-tuning (SFT) shows no such rise but a slight broadening,

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.

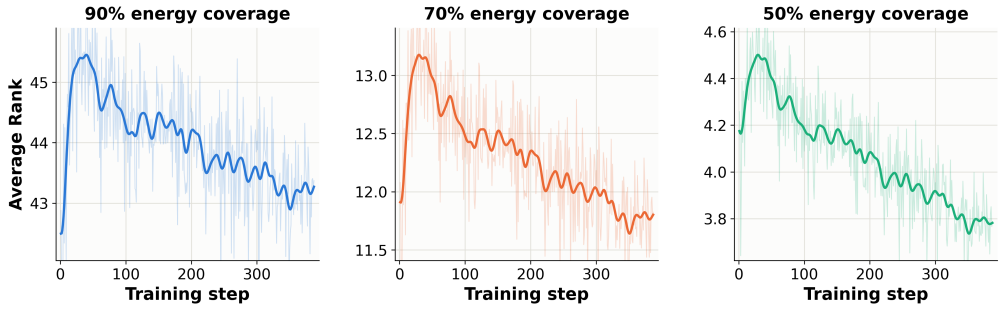


Figure 1: The rank $r(\varepsilon)$ needed to capture a fixed fraction ε of gradient energy (Qwen3-1.7B, GRPO, AdamW). After a brief early transient, $r(\varepsilon)$ declines steadily. Curves are lightly smoothed.

and replacing the reward with a random one, leaving rollouts and configuration intact, removes most of the increase, indicating that the trend is not explained solely by on-policy sampling.

This suggests that rank in RLVR is not merely a compression parameter but a quantity characterizing the evolving optimization geometry. Yet a concentrating spectrum need not imply a changing *useful* rank, since a large enough fixed rank could cover the whole run. To test this, we track the rank $r(\epsilon)$ needed to capture a fixed fraction ϵ of gradient energy during RLVR (Fig. 1). After a brief transient, $r(\epsilon)$ declines steadily. The effective dimensionality of the update, not merely the shape of the spectrum, evolves during training. Indeed, for RLVR gradients a small rank already suffices to cover the vast majority of the update subspace, and this coverage is attained at a shrinking rank as training proceeds. This motivates an optimizer that infers its update rank from the observed spectrum rather than prescribing it externally.

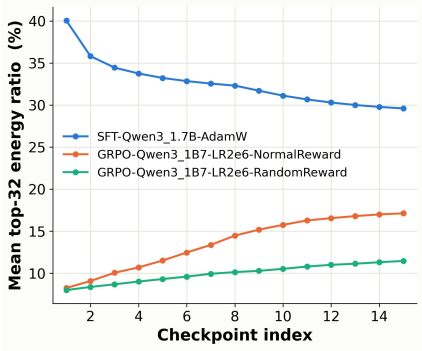


Figure 2: Top-32 energy ratio of gradient via RLVR (Qwen3-1.7B). We emphasize the trends, not absolute levels.

Actually, existing optimizers do not exploit this. Low-rank methods such as GaLore (Zhao et al., 2024) treat low rank as a compression device for memory-efficient pretraining, with an externally fixed rank; preconditioning and orthogonalization methods (*e.g.,* Shampoo (Gupta et al., 2018) and Muon (Jordan et al., 2024)) adapt during training but do not use the evolving spectral complexity of the gradient to set the dimensionality of the update subspace. The issue is thus not whether the gradient is low-rank, but where the rank comes from: a fixed small rank truncates non-negligible components when the spectrum broadens, an overly large one wastes capacity when it is sharply concentrated, and a constant or preset schedule sets the rank independently of the gradient. We present a detailed taxonomy of optimizer families on this axis in Sec. 4.4 (Table 1).

We introduce **Ares** (**A**daptive **R**ank-**E**ffective **S**pectral Optimization), which infers the update rank from the gradient's instantaneous spectral energy distribution, smoothly tracks its evolution, and confines the gradient signal driving each update to the resulting low-rank subspace, using randomized subspace iteration (Halko et al., 2011) to avoid a full SVD. Across LLMs at various scales, Ares improves average accuracy over widely used optimizers, with gains that hold across RLVR algorithms and extend beyond mathematics, and ablations confirm that each component contributes. Our contributions are as follows,

• **Spectral characterization of RLVR gradients:** we identify a persistent concentration of gradient energy in the leading singular components during RLVR training, with a declining entropy-based effective rank, which we do not observe under SFT or random-reward controls in the same setup.

• **A spectral-driven adaptive optimizer:** we propose Ares, which estimates effective rank from gradient energy distributions and dynamically adjusts its update subspace during RLVR training, rather than fixing the rank or following a preset schedule.

• **Performance improvement:** we validate this spectral-driven principle across model scales, downstream tasks, and RLVR algorithms, and isolate the contributing components through ablations.

2 Related Work

RLVR for LLMs. RLVR has become the dominant paradigm for post-training LLMs, first for aligning models with human preferences (Ouyang et al., 2022; Schulman et al., 2017), and recently for eliciting strong reasoning through verifiable rewards (DeepSeek-AI, 2025; Lambert et al., 2024). In this RLVR setting, a large body of work has focused on the policy-optimization objective. PPO (Schulman et al., 2017) relies on a learned value model, which GRPO (Shao et al., 2024) removes by estimating advantages from sampled responses; subsequent variants such as REINFORCE++ (Hu, 2025) and DAPO (Yu et al., 2025) further simplify or stabilize this estimate. These methods reshape the training signal but leave the underlying optimizer untouched, typically inheriting AdamW (Loshchilov & Hutter, 2019) from SFT. In contrast, we study the optimization geometry that RLVR induces and design the optimizer around it.

Optimizers for LLM Training. Adam (Kingma & Ba, 2015) and AdamW (Loshchilov & Hutter, 2019) remain the default optimizers for training LLMs. A large body of work seeks to improve upon them, either by exploiting curvature through matrix preconditioning (*e.g.,* Shampoo (Gupta et al., 2018), SOAP (Vyas et al., 2025), and the diagonal second-order method Sophia (Liu et al., 2024)) or by orthogonalizing the update, as in Muon (Jordan et al., 2024), while others discover update rules through symbolic search (Chen et al., 2023). A complementary line targets memory efficiency: Adafactor (Shazeer & Stern, 2018) and CAME (Luo et al., 2023) factor the optimizer state, and GaLore (Zhao et al., 2024) projects gradients onto a low-rank subspace, motivated by the fact that training progresses within a low-dimensional subspace (Gur-Ari et al., 2018). However, these methods treat rank as an externally fixed compression budget rather than a property of the gradient, and none is designed for RLVR. Our optimizer infers the update rank from the evolving spectral structure of RLVR gradients, adapting the subspace dimension online throughout RLVR.

3 Preliminary

RLVR for LLMs. RLVR fine-tunes a language model policy π_θ so that it produces correct solutions to problems drawn from a data distribution $\mathcal{D}$. Given a prompt $x \sim \mathcal{D}$, the policy samples a response $y \sim \pi_\theta(\cdot \mid x)$, which is scored by a rule-based verifier $r(x, y) \in \{0, 1\}$ that returns 1 only when y reaches the correct final answer. The training objective is to maximize the expected reward while staying close to a reference policy π_{ref}:

$$\max_\theta \; \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_\theta(\cdot|x)}\big[\, r(x,y)\,\big] - \beta\, \mathbb{D}_{\mathrm{KL}}\big(\pi_\theta(\cdot \mid x) \,\|\, \pi_{\text{ref}}(\cdot \mid x)\big), \tag{1}$$

where β controls the strength of the KL regularization. Since r is non-differentiable, this objective is optimized with policy-gradient methods. In this work we mainly adopt a popular algorithm GRPO (Shao et al., 2024), which samples a group of G responses $\{y_i\}_{i=1}^G$ per prompt and estimates a group-relative advantage $\hat{A}_i = (r_i - \mathrm{mean}(\{r_j\}))/\mathrm{std}(\{r_j\})$, avoiding a separately learned value model. The corresponding gradient reweights the log-likelihood of each sampled token by its advantage, $\nabla_\theta \mathcal{J} \propto \mathbb{E}\big[\hat{A}_i \, \nabla_\theta \log \pi_\theta(y_i \mid x)\big]$, so that positive-advantage responses are reinforced and negative ones suppressed. Furthermore, importance sampling and clipping are employed to reduce the variance caused by off-policy sampling and enhance the stability of RLVR training.

AdamW Optimizer. Given the estimated gradient g_t of the loss with respect to a parameter at step t, AdamW (Loshchilov & Hutter, 2019) maintains exponential moving averages of the first and second moments, $m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$ and $v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$, and applies a bias-corrected, per-coordinate adaptive update with decoupled weight decay,

$$\theta_t = \theta_{t-1} - \eta \left(\frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} + \lambda\, \theta_{t-1} \right), \tag{2}$$

where $\hat{m}_t$ and $\hat{v}_t$ are the bias-corrected moments, η is the learning rate, and λ the weight-decay coefficient. AdamW is the default optimizer for language-model training and is inherited essentially unchanged into RLVR post-training, which serves as the primary baseline in this paper.

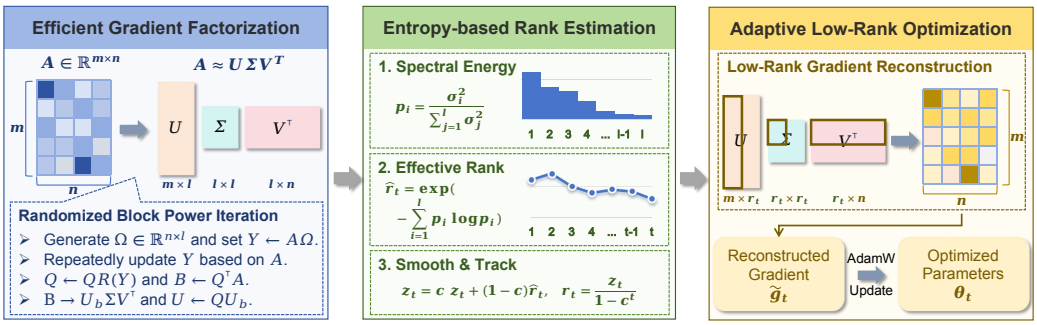


Figure 3: Ares overview. For 2-D tensors, Ares first decomposes the gradients using an approximate algorithm. Then, Ares estimates the effective rank based on the spectral energy and the rank momentum. Next, Ares uses the effective subspace to reconstruct the gradients, which is utilized to update parameters. 1-D tensors will be updated via standard AdamW. The pseudo code is in Appendix B.

4 Approach

As RLVR gradients progressively concentrate in leading singular components, their effective update dimensionality declines throughout training, motivating **Ares** to adapt its update rank directly to the observed spectrum. At each step, Ares (1) factorizes the gradient to identify its leading singular subspace (Sec. 4.1), (2) estimates and smoothly tracks the effective rank (Sec. 4.2), and (3) updates within the resulting low-rank subspace (Sec. 4.3). Fig. 3 illustrates the overall framework.

4.1 Efficient Gradient Factorization

To estimate the effective rank and remove noise during RLVR, Ares only needs the leading singular subspace at each step, yet dense SVD computes the entire spectrum at a cost quadratic in the smaller dimension n. Since RLVR gradients concentrate most of their energy in a small number of leading components (Fig. 1), resolving the low-energy tail is unnecessary. Ares therefore uses randomized block power iteration (RBPI) (Halko et al., 2011) to extract the leading subspace directly, with cost governed by the target rank rather than the ambient dimension.

Given a gradient matrix $A \in \mathbb{R}^{m\times n}$, a target rank k, an oversampling parameter p, and a number of power iterations q, RBPI draws a Gaussian test matrix $\Omega \in \mathbb{R}^{n\times \ell}$ with block size $\ell = k+p$ and forms the sketch $Y = A\Omega$. It then alternately applies $A^\top$ and A with a QR re-orthonormalization after each application, repeated q times, so that the resulting block behaves like $(AA^\top)^q A\,\Omega$. Since $(AA^\top)^q A$ shares the singular vectors of A but has singular values σ_i^{2q+1}, this power scheme amplifies the separation between the leading and trailing spectrum before the subspace is extracted. Orthonormalizing the block yields a basis Q, and projecting A onto it gives the small matrix $B = Q^\top A$, whose SVD is cheap; composing its left factors with Q recovers approximate singular vectors and values of A. The procedure outputs a factorization with left factor $U \in \mathbb{R}^{m\times \ell}$, singular values $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_\ell)$, and right factor $V \in \mathbb{R}^{n\times \ell}$ (so that $A \approx U\Sigma V^\top$), which Sec. 4.2 uses to estimate the effective rank.

This construction replaces the $O(mn^2)$ cost of a dense SVD with $O(q\,mn\ell)$, an advantage that is substantial precisely when $q(k+p) \ll n$, the regime Ares operates in since transformer weight gradients have n on the order of 10^3 to 10^4 while the target rank is a few tens. The accuracy of the approximation is likewise favorable: RBPI is exact when A is genuinely rank-k, and in general its projection error is bounded by a multiplicative factor times the optimal rank-k error σ_{k+1}, with the factor tightening as q grows and as the spectral gap widens. The spectral concentration of RLVR gradients established in Sec. 1 supplies exactly such a gap, so a moderate q and a small oversample let RBPI track the leading subspace closely. The full cost and analysis is in Appendix C.

4.2 Spectral Energy-based Rank Estimation

The factorization exposes the gradient spectrum, but it does not by itself specify how many components the update should retain. Sec. 1 shows that this number is neither constant across training nor

well described by the algebraic rank, which is almost always full for a dense gradient; what evolves is the concentration of energy. Ares therefore reads the rank off the spectral energy distribution and tracks its evolution smoothly over training.

Spectral Energy Modeling. We characterize the spectrum by the energy carried by each singular direction rather than by the presence or absence of components. Writing $\sigma_1 \geq \cdots \geq \sigma_{k+p} \geq 0$ for the singular values returned by the factorization, we form the normalized energy distribution

$$p_i = \frac{\sigma_i^2}{\sum_j \sigma_j^2}, \tag{3}$$

so that p_i measures the fraction of gradient energy along the i-th direction. Since RBPI returns only the leading $\ell = k + p$ components, the normalization is over the retained subspace rather than the full gradient, so $\hat{r}_t$ below is an effective rank measured *within the target-rank budget* k. We then summarize the concentration of this distribution by its Shannon entropy and convert it into an effective rank (Roy & Vetterli, 2007) through

$$H_t = -\sum_i p_i \log p_i, \qquad \hat{r}_t = \exp(H_t). \tag{4}$$

The $\hat{r}$ admits a direct interpretation as the number of directions over which the energy is effectively spread: when the energy is uniform across the ℓ retained directions the entropy is maximal and $\hat{r}_t$ attains its upper bound ℓ, whereas when the energy collapses onto a few leading components the entropy shrinks and $\hat{r}_t$ becomes small. As it depends on the instantaneous spectrum, $\hat{r}_t$ is a measure of the current spectral complexity of the gradient and moves as the spectrum evolves, which is precisely the behavior of the declining $r(\epsilon)$ in Fig. 1.

Rank Adaptation. The per-step estimate $\hat{r}_t$ inherits the stochasticity of the sampled gradient, and using it directly would cause the dimension of the update subspace to fluctuate from step to step. Ares therefore smooths the estimate with an exponential moving average across steps. Let c denote the smoothing coefficient and z_t an uncorrected EMA accumulator initialized at $z_0 = 0$. We update

$$z_t = c\, z_{t-1} + (1-c)\,\hat{r}_t, \qquad r_t = \mathrm{clip}\Big(\mathrm{round}\Big(\frac{z_t}{1-c^{\,t}}\Big),\ 1,\ k\Big), \tag{5}$$

where the accumulator z_t is kept separate from the tracked rank so that the division by $1 - c^t$ implements the standard EMA bias correction from a zero start, $\mathrm{round}(\cdot)$ maps the real-valued effective rank to an integer index, and the clipping keeps the rank within $[1, k]$. Because r_t can only decrease from the ceiling k, Ares performs adaptive truncation within a preset maximum-rank budget rather than discovering a rank above k. The tracked rank r_t thus follows the concentration of gradient energy continuously over training while remaining stable against per-step noise. This is the sense in which the rank in Ares is inferred from the observed spectrum and tracked as it evolves, in contrast to a fixed rank as in GaLore, the distinction made explicit in Sec. 4.4.

4.3 Adaptive Low-Rank Optimization

Based on r_t, the final stage forms the parameter update within the selected subspace. Ares reconstructs a low-rank gradient by truncating the factorization of Sec. 4.1 to its leading r_t components,

$$\tilde{g}_t = U_{:,\,1:r_t}\, \Sigma_{1:r_t}\, V^{\top}_{1:r_t,\,:}, \tag{6}$$

which is the rank-r_t truncated SVD reconstruction: it projects the raw gradient onto the spectral subspace that carries its dominant energy, weighted by the singular values $\Sigma_{1:r_t} = \mathrm{diag}(\sigma_1, \ldots, \sigma_{r_t})$, and discards the low-energy directions. Retaining Σ preserves the energy-weighting of the leading directions, so the reconstruction is a faithful truncation of the gradient rather than an unweighted projection. The reconstructed gradient then drives an adaptive update in the manner of the AdamW rule of Sec. 3: Ares maintains bias-corrected first- and second-moment estimates of $\tilde{g}_t$, forms the per-coordinate adaptive step, and applies it to the parameter together with decoupled weight decay,

$$\theta_t = (1 - \eta\lambda)\,\theta_{t-1} - \eta\, s\, \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}, \tag{7}$$

where η is the learning rate, λ the weight-decay coefficient, and s a scale factor for the low-rank update. Operating on $\tilde{g}_t$ confines the update to informative spectral directions, avoiding wasted capacity on the low-energy tail.

Table 1: Comparison of Ares with representative optimizers used in large-model training. We distinguish three properties: whether the method restricts the update to a *low-rank* subspace, *how* the rank/update subspace is determined, and whether that determination is *spectral-driven*, *i.e.,* inferred from the gradient's spectral energy distribution during training. Ares infers the *dimensionality* of the update subspace from the evolving gradient spectrum.

Optimizer	**Update rule**	**Low-rank**	**Rank**	**Spectral-driven**
SGD	First-order momentum	No	—	No
RMSprop	Diagonal adaptive	No	—	No
Adam (Kingma & Ba, 2015)	Diagonal adaptive	No	—	No
AdamW (Loshchilov & Hutter, 2019)	Diagonal adaptive	No	—	No
Adafactor (Shazeer & Stern, 2018)	Factored second moment	No	—	No
Lion (Chen et al., 2023)	Sign momentum	No	—	No
Sophia (Liu et al., 2024)	Diagonal Hessian precond.	No	—	No
Shampoo (Gupta et al., 2018)	Full-matrix precond.	No	—	No
SOAP (Vyas et al., 2025)	Shampoo in Adam eigenbasis	No	—	No
Muon (Jordan et al., 2024)	Orthogonalized momentum	No	—	Direction only
GaLore (Zhao et al., 2024)	Low-rank projection	Yes	Constant r	No
Rank schedule	Low-rank projection	Yes	Preset $r(t)$	No
Ares (ours)	Low-rank, spectral-driven	Yes	Spectrum	Yes

This spectral treatment applies only to parameters with a two-dimensional matrix structure, for which a singular spectrum is defined. Ares thus routes by shape: two-dimensional weight matrices receive the low-rank spectral update above, while one-dimensional parameters take a standard AdamW step. The full procedure is summarized in Appendix B.

4.4 Summarization and Discussion

We summarize Ares in Algorithm 1. At each step, Ares factorizes gradients with a randomized block power iteration, converts the resulting spectral energy distribution into an effective rank through its entropy, smoothly tracks this rank across steps, and forms an adaptive update confined to the corresponding low-rank subspace, while one-dimensional parameters retain a standard AdamW step. Its central contribution is that the update rank is neither fixed nor scheduled but *inferred from the observed gradient spectrum* and tracked as it evolves, so the update dimensionality follows the energy concentration, at a per-step cost governed by the target rank rather than the ambient dimension.

Table 1 places Ares among representative optimizers along three axes: whether the update is restricted to a low-rank subspace, how the rank is determined, and whether that determination is spectral-driven. First-order, diagonal adaptive, and preconditioned methods (*e.g.,* SGD, AdamW, Sophia, SOAP) all apply full-dimensional updates, adapting only the step size or a preconditioner, and Muon adapts the update *direction* but leaves its dimensionality unconstrained. Among low-rank methods, GaLore fixes the rank to a constant and a rank schedule varies it along a preset curve, so in both cases the rank is chosen independently of the observed gradient. Ares is thus the one method in Table 1 that is low-rank *and* spectral-driven, with r_t read online from the spectrum and tracked as it evolves. Concurrent work has begun to study low-rank and spectral structure specifically in RLVR (Chen et al., 2026; Fan et al., 2026); relative to these, our distinguishing property is that the rank is neither fixed nor scheduled but inferred from the gradient's spectral energy. This is precisely the property motivated in Sec. 1 and constitutes the core contribution of this work: matching the optimizer to the evolving gradient geometry rather than to a fixed rank.

5 Experiment

5.1 Experiment Settings

Dataset. We conduct RLVR on the DAPO dataset (Yu et al., 2025), a large-scale collection of competition-level mathematics problems paired with verifiable answers. To assess generalization, we evaluate on seven widely used mathematical reasoning benchmarks that span a range of dif-

Table 2: Accuracy of LLMs trained via GRPO with various optimizers. "Avg." refers to average accuracy. The best is in bold and the second is underlined.

Optimizer	AIME24	AIME25	AIME26	AMC23	MATH500	Minerva	OlyBench	Avg.
				Qwen3-1.7B-Base				
AdamW	7.9	4.2	5.0	37.8	59.2	15.5	25.7	22.2
SGD	0.4	0.0	0.4	2.8	6.3	2.9	1.3	2.0
RMSprop	0.0	0.0	0.0	13.1	33.5	4.8	9.6	8.7
Adam	7.5	<u>6.3</u>	4.6	<u>40.0</u>	<u>59.3</u>	15.3	25.1	22.6
GaLore	<u>8.3</u>	<u>6.3</u>	<u>6.3</u>	37.8	58.9	<u>16.9</u>	<u>26.1</u>	<u>22.9</u>
Muon	0.4	0.4	0.0	3.4	7.8	2.8	1.4	2.3
Ares (Ours)	**15.0**	**10.8**	**10.0**	**43.4**	**61.6**	**20.4**	**29.4**	**27.2**
				Qwen3-4B-Base				
AdamW	<u>21.3</u>	16.3	13.8	<u>60.9</u>	<u>71.9</u>	<u>27.4</u>	<u>39.1</u>	<u>35.8</u>
SGD	0.8	0.4	0.4	6.6	14.5	5.7	3.8	4.6
RMSprop	2.5	0.8	2.5	13.1	38.7	7.6	12.9	11.2
Adam	18.3	<u>16.7</u>	<u>15.4</u>	57.2	70.6	23.3	36.8	34.0
GaLore	15.8	13.3	12.9	55.9	70.8	23.4	37.2	32.8
Muon	1.3	0.4	0.4	8.8	17.5	6.6	5.4	5.8
Ares (Ours)	**25.4**	**17.5**	**20.0**	**65.0**	**73.4**	**28.2**	**40.3**	**38.5**
				Qwen3-8B-Base				
AdamW	<u>25.8</u>	19.6	19.2	<u>68.4</u>	**75.0**	<u>27.7</u>	<u>42.6</u>	<u>39.8</u>
SGD	0.8	1.3	0.8	10.9	21.8	8.0	7.1	7.2
RMSprop	2.0	3.3	1.3	18.8	24.7	10.7	12.2	10.4
Adam	23.3	19.6	17.5	67.8	74.1	22.7	40.9	38.0
GaLore	25.4	<u>20.8</u>	<u>20.4</u>	66.9	73.2	24.5	40.3	38.8
Muon	2.1	3.3	2.5	15.6	23.2	7.1	8.5	8.9
Ares (Ours)	**28.8**	**22.5**	**21.7**	**73.8**	<u>74.9</u>	**29.4**	**43.4**	**42.1**

ficulty: AIME24 (AIME2024, 2024), AIME25 (AIME2025, 2025), AIME26 (AIME2026, 2026), AMC23 (AMC2023, 2023), MATH500 (Lightman et al., 2024), Minerva Math (Lewkowycz et al., 2022), and OlympiadBench (OlyBench) (He et al., 2024). Following common practice, we report the average accuracy over 8 sampled responses per problem to reduce evaluation variance.

Baselines. We compare Ares against a representative set of optimizers commonly used for LLM training. These include (1) standard first-order and adaptive optimizers, *e.g.,* SGD, RMSprop, Adam (Kingma & Ba, 2015), and AdamW (Loshchilov & Hutter, 2019); and (2) a low-rank optimizer, GaLore (Zhao et al., 2024), which projects gradients onto a subspace of externally fixed rank, and Muon (Jordan et al., 2024), which orthogonalizes the momentum update. All baselines are trained under an identical RLVR pipeline. We privode the implementation details in Appendix A.

5.2 Main Results

Table 2 reports the performance of different optimizers with GRPO. We summarize four findings.

First, Ares delivers consistent and substantial gains across model scales. Ares attains the best average accuracy at every scale, *i.e.,* 27.2, 38.5, and 42.1 on the 1.7B, 4B, and 8B models, and ranks first on the large majority of the seven individual benchmarks. It improves over the widely used AdamW by 5.0, 2.7, and 2.3 points. The improvement is therefore not confined to a particular setting but holds broadly across benchmarks and model sizes.

Second, the advantage is stable as the model scales up. Rather than shrinking with capacity, the lead of Ares persists from 1.7B to 8B, and it remains the top method at every scale. This indicates that the benefit of matching the optimizer to the RLVR gradient geometry does not vanish as the base model grows stronger, so the approach scales gracefully rather than being an artifact of small models.

Third, naively exploiting low-rank structure is insufficient to capture the gradient characteristics of RLVR. GaLore, which projects gradients onto a subspace of externally fixed rank, is merely on par with AdamW and is in fact worse at the 4B scale (32.8 vs. 35.8). Imposing a low-rank constraint

Table 3: Ablation study of Ares on Qwen3-1.7B-Base on mathematical tasks. "Avg." refers to average accuracy. Further results of various models can be found in Table 4.

Optimizer	AIME24	AIME25	AIME26	AMC23	MATH500	Minerva	OlyBench	Avg.
Ares (Ours)	15.0	10.8	10.0	43.4	61.6	20.4	29.4	27.2
w/o Rank Momentum	10.8	7.5	5.8	41.9	57.9	18.2	26.7	24.1
w/o Dynamic Rank	12.9	9.2	5.0	48.1	61.9	19.7	29.5	26.6
w/o Low Rank	0.0	0.0	0.4	4.4	9.1	2.9	2.0	2.7
w/o Spectral Normalize	10.8	6.7	5.0	48.1	61.9	20.0	28.0	25.8
w/o Adaptive LR	10.0	5.0	4.2	39.4	60.5	17.4	25.9	23.2

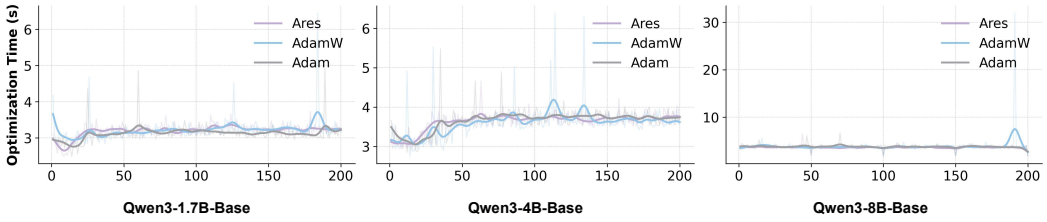


Figure 4: The time to update model parameters expended by various optimizers.

alone is thus not enough: because the spectral concentration of RLVR gradients evolves during training, the update rank must be inferred from the gradient spectrum and adapted online rather than fixed in advance, which is precisely what distinguishes Ares from GaLore.

Finally, the degree to which the optimizer matches RLVR dynamics is critical to final performance. The optimizers differ markedly from one another: SGD, RMSprop, and Muon are consistently weak across all three scales, collapsing to near-zero or single-digit accuracy, whereas AdamW and GaLore are more stable. Against this backdrop, Ares improves further at every scale. Taken together, these gaps show that the optimizer is not an interchangeable building block in RLVR; its design must be tailored to the distinctive and dynamically evolving gradient geometry that RLVR induces.

5.3 Detailed Analysis

Ablation Study. Table 3 isolates each component of Ares. Removing any single component lowers accuracy, so the modules are complementary. The low-rank update is the core: *w/o Low Rank* collapses accuracy to 2.7 and this collapse persists even with the other mechanisms active, as adjusting the learning rate or fixing $\sigma = 1$ (an orthogonalized step in the spirit of Muon) brings no benefit without it; confining the update to the low-rank spectral subspace is thus the foundation that makes every other component useful. Dynamic rank and momentum, in turn, act as one mechanism: *w/o Dynamic Rank* (26.6) alone appears to gain little, because a rank estimated from noisy on-policy gradients fluctuates and destabilizes the update subspace, but momentum smooths this estimate and stabilizes the subspace, so that *w/o Momentum* drops more sharply to 24.1. Their combination turns a fluctuating rank into a reliable gain, showing that the pieces reinforce one another rather than forming a loose collection of independently useful tricks.

Analysis of Computational Consumption. Fig. 4 reports the per-step time each optimizer spends updating the parameters. Although Ares performs extra computation over a plain adaptive step, estimating the spectral energy and forming the low-rank subspace, this is kept cheap by using randomized subspace iteration (Halko et al., 2011) to extract only the leading r components rather than a full SVD. As a result, its update time stays close to standard optimizers and far below an exact spectral method, confirming that the spectral-driven design adds only low-order overhead. The low-rank factorization also opens a path to lower resource usage, since the first- and second-moment states could in principle be maintained in a factored form (Zhao et al., 2024). We stress, however, that our current implementation does not yet realize this saving, which maintains dense moment ten-

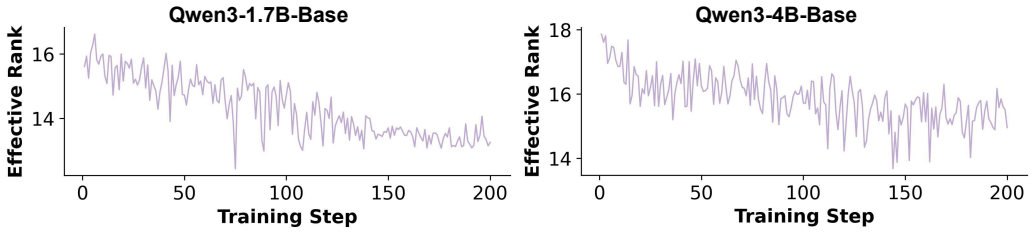


Figure 5: The dynamics of the effective rank of the gradient during the RLVR procedure.

sors as in AdamW, so the present contribution is update quality rather than optimizer-state memory, which we leave for future work.

Dynamics of Energy. Fig. 5 tracks the effective rank of the gradient, estimated from its spectral energy distribution over the RLVR. First, the effective rank declines steadily as training proceeds. The gradient energy thus concentrates into progressively fewer directions during RLVR, and the effective dimensionality of the useful update shrinks over time. This directly corroborates the low-rank characterization of Sec. 1 on the very quantity Ares acts on, and it is what justifies inferring the update rank from the spectrum and letting it decay online rather than fixing it in advance. Second, the larger model operates at a consistently higher effective rank: the 4B curve sits above the 1.7B curve throughout training, both at initialization and after convergence. The larger model spreads its gradient energy over more effective directions, suggesting that greater capacity lets it explore and exploit a richer set of useful update directions and thereby encode more knowledge. This also carries a practical implication for Ares: because the appropriate rank grows with model capacity, inferring it from the observed spectrum, is what allows the same optimizer to adapt across model scales.

Effect of Different Rank Adaptation Strategies. Given the estimated spectral energy, Ares still needs a rule to translate it into a concrete rank. We compare three strategies in Fig. 6: *Exp* maps energy to rank exponentially, *Linear* in direct proportion, and *Threshold* sets the rank at the smallest value whose cumulative energy reaches a fixed fraction, an abrupt cutoff. The exponential mapping used in Ares performs best overall, attaining the top or near-top accuracy at both scales and never being clearly beaten, which makes it a robust default. Smaller models are markedly more sensitive to the choice: the three strategies span a wide range on Qwen3-1.7B but are nearly indistinguishable among the smooth mappings on Qwen3-4B, as a smaller model has little margin to absorb a poorly chosen update subspace. *Threshold* is consistently worst, since fixed energy thresholds over-admit weak directions under long-tailed spectra, whereas a smooth mapping adapts rank continuously to spectral shape, motivating Ares's exponential form.

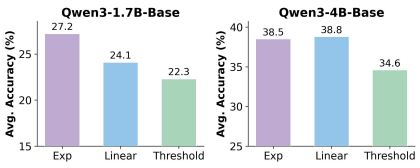


Figure 6: Accuracy of various adaptation methods. Full results in Table 5

6 Conclusion

In this work, we revisited optimizer design for RLVR and show that its gradients exhibit a distinctive, evolving spectral geometry: energy concentrates progressively in leading singular components, driving the effective update rank downward, in contrast to the SFT and random-reward controls we study. Based on this observation, we introduced **Ares**, which automatically infers and smoothly adapts its update rank from the gradient spectrum while confining the gradient signal of each update to the resulting low-rank subspace. Extensive experiments across model scales, tasks, and RLVR algorithms demonstrated consistent gains over widely used optimizers, with ablations supporting the contribution of each component. Our findings pointed to a broader principle: effective optimizers should adapt to the gradient geometry induced by the training paradigm, rather than inherit assumptions from supervised training.

AI USE STATEMENT

We used an AI assistant solely to polish the writing and check for grammatical errors in the manuscript. It was not involved in any other part of this work, including the research ideation, method design, implementation, experiments, or analysis of the results. All technical contributions and scientific claims are the authors' own.

ETHICS STATEMENT

This work studies optimization for reinforcement learning with verifiable rewards and does not involve human subjects, personally identifiable information, or sensitive data. All experiments are conducted on publicly available mathematical reasoning datasets and open-weight base models, used in accordance with their respective licenses. Our contribution is a general-purpose optimizer; it does not target any harmful application, and we are not aware of ethical concerns beyond those already associated with training large language models, such as the computational cost and energy consumption of RLVR post-training. We report the compute used in our experiments to promote transparency in this regard.

REPRODUCIBILITY STATEMENT

We have taken several steps to ensure the reproducibility of our results. The RLVR objective and the AdamW baseline are formalized in Section 3, and our method is described in full in Section 4, with the complete algorithm given as pseudocode in Appendix B and source code in the supplementary material. All datasets, base models, benchmarks, and evaluation protocols are specified in Section 5.1, and the training configuration and full set of hyperparameters are provided in Section 5.1 and Appendix A. All experiments use publicly available datasets and open-weight models, and we will release our code to facilitate reproduction of the reported results.

REFERENCES

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting reinforce-style optimization for learning from human feedback in llms. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pp. 12248–12267. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.ACL-LONG.662. URL https://doi.org/10.18653/v1/2024.acl-long.662.

AIME2024. Aime2024, 2024. URL https://huggingface.co/datasets/HuggingFaceH4/aime_2024.

AIME2025. Aime2025, 2025. URL https://huggingface.co/datasets/opencompass/AIME2025.

AIME2026. Aime2026, 2026. URL https://huggingface.co/datasets/MathArena/aime_2026.

AMC2023. Amc2023, 2023. URL https://huggingface.co/datasets/math-ai/amc23.

Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, and Quoc V. Le. Symbolic discovery of optimization algorithms. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/9a39b4925e35cf447ccba8757137d84f-Abstract-Conference.html.

Zhipeng Chen, Tao Qian, Wayne Xin Zhao, and Ji-Rong Wen. Low-rank optimization trajectories modeling for LLM RLVR acceleration. *CoRR*, abs/2604.11446, 2026. doi: 10.48550/ARXIV.2604.11446. URL https://doi.org/10.48550/arXiv.2604.11446.

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *CoRR*, abs/2501.12948, 2025. doi: 10.48550/ARXIV.2501.12948. URL https://doi.org/10.48550/arXiv.2501.12948.

Chongyu Fan, Gaowen Liu, Mingyi Hong, Ramana Rao Kompella, and Sijia Liu. Rethinking muon beyond pretraining: Spectral failures and high-pass remedies for VLA and RLVR. *CoRR*, abs/2605.19282, 2026. doi: 10.48550/ARXIV.2605.19282. URL https://doi.org/10.48550/arXiv.2605.19282.

Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization. In Jennifer G. Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1837–1845. PMLR, 2018. URL http://proceedings.mlr.press/v80/gupta18a.html.

Guy Gur-Ari, Daniel A. Roberts, and Ethan Dyer. Gradient descent happens in a tiny subspace. *CoRR*, abs/1812.04754, 2018. URL http://arxiv.org/abs/1812.04754.

Nathan Halko, Per-Gunnar Martinsson, and Joel A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Rev.*, 53(2): 217–288, 2011. doi: 10.1137/090771806. URL https://doi.org/10.1137/090771806.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. Olympiadbench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pp. 3828–3850. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.ACL-LONG.211. URL https://doi.org/10.18653/v1/2024.acl-long.211.

Jian Hu. REINFORCE++: A simple and efficient approach for aligning large language models. *CoRR*, abs/2501.03262, 2025. doi: 10.48550/ARXIV.2501.03262. URL https://doi.org/10.48550/arXiv.2501.03262.

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

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tülu 3: Pushing frontiers in open language model post-training. *CoRR*, abs/2411.15124, 2024. doi: 10.48550/ARXIV.2411.15124. URL https://doi.org/10.48550/arXiv.2411.15124.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. Solving quantitative reasoning problems with language models, 2022.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

Hong Liu, Zhiyuan Li, David Leo Wright Hall, Percy Liang, and Tengyu Ma. Sophia: A scalable stochastic second-order optimizer for language model pre-training. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=3xHDeA8Noi.

Shih-Yang Liu, Xin Dong, Ximing Lu, Shizhe Diao, Peter Belcak, Mingjie Liu, Min-Hung Chen, Hongxu Yin, Yu-Chiang Frank Wang, Kwang-Ting Cheng, Yejin Choi, Jan Kautz, and Pavlo Molchanov. GDPO: group reward-decoupled normalization policy optimization for multi-reward RL optimization. *CoRR*, abs/2601.05242, 2026. doi: 10.48550/ARXIV.2601.05242. URL https://doi.org/10.48550/arXiv.2601.05242.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Yang Luo, Xiaozhe Ren, Zangwei Zheng, Zhuo Jiang, Xin Jiang, and Yang You. CAME: confidence-guided adaptive memory efficient optimization. In Anna Rogers, Jordan L. Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pp. 4442–4453. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.ACL-LONG.243. URL https://doi.org/10.18653/v1/2023.acl-long.243.

OpenAI. Openai o1 system card. *CoRR*, abs/2412.16720, 2024. doi: 10.48550/ARXIV.2412.16720. URL https://doi.org/10.48550/arXiv.2412.16720.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/b1efde53be364a73914f58805a001731-Abstract-Conference.html.

Olivier Roy and Martin Vetterli. The effective rank: A measure of effective dimensionality. In *15th European Signal Processing Conference, EUSIPCO 2007, Poznan, Poland, September 3-7, 2007*, pp. 606–610. IEEE, 2007. URL https://ieeexplore.ieee.org/document/7098875/.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *CoRR*, abs/2402.03300, 2024. doi: 10.48550/ARXIV.2402.03300. URL https://doi.org/10.48550/arXiv.2402.03300.

Noam Shazeer and Mitchell Stern. Adafactor: Adaptive learning rates with sublinear memory cost. In Jennifer G. Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4603–4611. PMLR, 2018. URL http://proceedings.mlr.press/v80/shazeer18a.html.

Qwen Team. Qwen3 technical report. *CoRR*, abs/2505.09388, 2025. doi: 10.48550/ARXIV.2505.09388. URL https://doi.org/10.48550/arXiv.2505.09388.

Nikhil Vyas, Depen Morwani, Rosie Zhao, Itai Shapira, David Brandfonbrener, Lucas Janson, and Sham M. Kakade. SOAP: improving and stabilizing shampoo using adam for language modeling. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=IDxZhXrpNf.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Juncai Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Ru Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou,

Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Yonghui Wu, and Mingxuan Wang. DAPO: an open-source LLM reinforcement learning system at scale. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025. URL http://papers.nips.cc/paper_files/paper/2025/hash/a4277440d50f1f15d2cb4c14f7e0c0d2-Abstract-Conference.html.

Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. Galore: Memory-efficient LLM training by gradient low-rank projection. In Ruslan Salakhutdinov, Zico Kolter, Katherine A. Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, volume 235 of *Proceedings of Machine Learning Research*, pp. 61121–61143. PMLR / OpenReview.net, 2024. URL https://proceedings.mlr.press/v235/zhao24s.html.

A Implementation Details

We build on the Qwen3-Base models (Team, 2025) at three scales, *i.e.,* 1.7B, 4B, and 8B, and train all optimizers with GRPO (Shao et al., 2024) under the same configuration. For each training step, we sample a group of 16 responses per prompt with a rollout batch size of 128 and a maximum generation length of 8192 tokens, and update the policy with a mini-batch size of 32. We tune the learning rate for each optimizer separately, searching over $\{1, 2, 5\}\times10^{-6}$ and $\{1, 2\}\times10^{-5}$ with the same budget for every optimizer (AdamW, GaLore, Muon, and Ares) and selecting by final training accuracy, and use 5×10^{-6} for Ares. For the low-rank optimizers, the rank is set to 64; for Ares, we initialize the rank at 64 and adapt it online from the gradient spectrum. For the Ares-specific hyperparameters we use RBPI oversampling $p = 10$ and power iterations $q = 20$ (Appendix C), rank-EMA smoothing $c = 0.9$, and scale $s = 1$, shared across all layers and model scales rather than tuned per layer; the rank-to-index mapping uses nearest-integer rounding. All models are trained for 300 steps.

B Pseudo Code of Ares

Algorithm 1 summarizes one optimizer step of Ares, following the notation of Sec. 4. Two-dimensional weights receive the spectral low-rank update (Sections 4.1–4.3), while one-dimensional parameters fall back to a standard AdamW step. State maintained across steps consists of the moment estimates (m, v) and the tracked rank r; the momentum coefficients (β_1, β_2) follow the AdamW convention of Sec. 3.

Algorithm 1: Ares: one optimizer step for a single parameter θ

Input: gradient g_t; step t; learning rate η, weight decay λ, scale s, epsilon ϵ; smoothing c; target rank k; RBPI parameters (p, q)

State: first/second moments m, v; uncorrected rank EMA accumulator z (init. $z \leftarrow 0$); tracked rank r (init. $r \leftarrow k$)

if θ *is a 2D weight matrix* **then**
- $U, \Sigma, V \leftarrow \mathrm{RBPI}(g_t,\, k,\, p,\, q)$ — `// leading singular subspace (Sec. 4.1)`
- $p_i \leftarrow \sigma_i^2 / \sum_j \sigma_j^2$; $H \leftarrow -\sum_i p_i \log p_i$; $\hat{r}_t \leftarrow \exp(H)$ — `// energy entropy (Sec. 4.2)`
- $z \leftarrow c\, z + (1-c)\, \hat{r}_t$; $r \leftarrow \mathrm{clip}\big(\mathrm{round}(z/(1-c^t)),\, 1,\, k\big)$ — `// smooth and track`
- $\tilde{g}_t \leftarrow U_{:,\,1:r}\, \Sigma_{1:r}\, V^\top_{1:r,\,:}$ — `// rank-r truncated-SVD gradient`

else
- $\tilde{g}_t \leftarrow g_t$ — `// 1D parameter: standard AdamW`

$m \leftarrow \beta_1 m + (1-\beta_1)\tilde{g}_t$; $v \leftarrow \beta_2 v + (1-\beta_2)\tilde{g}_t^2$ — `// moment updates`

$\hat{m} \leftarrow m/(1-\beta_1^t)$; $\hat{v} \leftarrow v/(1-\beta_2^t)$ — `// bias correction`

$s_\theta \leftarrow s$ if θ is 2D else 1 — `// low-rank scaling only for matrices`

$\theta \leftarrow (1-\eta\lambda)\,\theta - \eta\, s_\theta\, \hat{m}/(\sqrt{\hat{v}} + \epsilon)$ — `// decoupled weight decay + update`

return θ

C Analysis of the Randomized Block Power Iteration

At the core of Ares is a randomized block power iteration (RBPI) that estimates, at every optimizer step, the leading singular subspace of the (momentum-smoothed) gradient matrix $A \in \mathbb{R}^{m\times n}$. Rather than computing a full singular value decomposition (SVD), Ares uses RBPI as an approximation whose leading-order cost is governed by the target rank k rather than by the smaller ambient dimension $\min(m, n)$. This appendix analyzes (i) the computational cost of RBPI relative to a dense SVD, and (ii) the sense in which RBPI approximates the leading spectral subspace, together with the conditions under which the approximation is expected to be accurate.

Throughout, we denote the input matrix by $A \in \mathbb{R}^{m\times n}$, and without loss of generality we assume $m \geq n$ (the implementation transposes internally, so the analysis is symmetric in m and n). We write $r = \min(m, n)$ for the full rank, k for the target rank, p for the oversampling parameter (`oversample`, default $p = 10$), and q for the number of power iterations (`n_iter`, default $q = 20$). We set $\ell = k + p$ for the size of the sampled block. The singular values of A are $\sigma_1 \geq \sigma_2 \geq \cdots \geq$

$\sigma_r \geq 0$, and by the Eckart–Young theorem the best rank-k approximation error is $\|A - A_k\|_2 = \sigma_{k+1}$ and $\|A - A_k\|_F^2 = \sum_{j>k} \sigma_j^2$, where A_k is the truncation of the SVD to its top k components.

Before analyzing the two aspects above, we note that the routine is not a one-pass randomized SVD but rather Halko et al.'s *power scheme* (Halko et al., 2011): after the initial sketch $Y = A\Omega$, the loop alternately applies $A^\top$ and A with a QR re-orthonormalization after each application, repeated q times, so that the final block satisfies $Y_q \approx (AA^\top)^q A\,\Omega$. Since $(AA^\top)^q A$ has the same singular vectors as A but singular values σ_i^{2q+1}, the scheme amplifies the separation between the leading and trailing spectrum, which is the property the approximation analysis below relies on.

C.1 Time Complexity

Dense SVD. Computing the thin SVD $A = U\Sigma V^\top$ of a dense $m \times n$ matrix with $m \geq n$ costs

$$T_{\mathrm{SVD}} = \Theta\big(mn^2\big) \tag{8}$$

for the standard bidiagonalization-based pipeline. This is the cost model of the dense driver behind `torch.linalg.svd`; the key point for our purposes is that, for such a routine, the cost scales with the full smaller dimension n rather than with the target rank k, so requesting only the top $k \ll n$ components does not by itself reduce the work. (Some backends offer specialized truncated or approximate SVD drivers; we do not rely on their availability here.)

Randomized block power iteration. We account for RBPI step by step, following the implementation. Matrix–matrix products dominate; QR of an $m \times \ell$ (respectively $n \times \ell$) matrix costs $O(m\ell^2)$ (respectively $O(n\ell^2)$), and the small dense SVD of the $\ell \times n$ matrix B costs $O(n\ell^2)$.

- **Sketch** $Y = A\Omega$ with $\Omega \in \mathbb{R}^{n\times\ell}$: $O(mn\ell)$.
- **Power iterations** (q rounds): each round forms $A^\top Y$ and AY at $O(mn\ell)$ each, plus two QR factorizations at $O((m+n)\ell^2)$; total $O\big(q\,(mn\ell + (m+n)\ell^2)\big)$.
- **Orthonormalization** Q of Y: $O(m\ell^2)$.
- **Projection** $B = Q^\top A \in \mathbb{R}^{\ell\times n}$: $O(mn\ell)$.
- **Small SVD** of B: $O(n\ell^2)$.
- **Lift** $U = Q\,U_B$: $O(m\ell^2)$.

Collecting the terms, the total cost is

$$T_{\mathrm{RBPI}} = O\big((q+1)\,mn\ell + (q+1)(m+n)\ell^2\big)\,. \tag{9}$$

When $\ell \leq n \leq m$ we have $(m+n)\ell^2 \leq 2m\ell^2 \leq 2mn\ell$, so the $mn\ell$ terms dominate and the bound simplifies to

$$T_{\mathrm{RBPI}} = O(q\,mn\ell)\,, \qquad \ell = k + p. \tag{10}$$

Rank-dependent cost and the resulting ratio. Treating the two costs as $T_{\mathrm{SVD}} = \Theta(mn^2)$ and $T_{\mathrm{RBPI}} = \Theta(q\,mn\ell)$ under the assumption $\ell \leq n$ (so that the leading terms above are indeed dominant), the leading-order work ratio is

$$\frac{T_{\mathrm{SVD}}}{T_{\mathrm{RBPI}}} = \Theta\!\left(\frac{n}{q\,(k+p)}\right). \tag{11}$$

The essential distinction is that the dense SVD grows quadratically in the smaller ambient dimension n, whereas the dominant matrix products in RBPI grow only linearly in the block size $\ell = k + p$. In other words, RBPI trades a dependence on n^2 for a dependence on $q(k+p)\,n$, and the asymptotic advantage becomes substantial precisely when $q(k+p) \ll n$. This is the regime Ares targets: gradient matrices of transformer weight layers have n on the order of 10^3–10^4, while the target rank is a few tens (initial rank 64 in our experiments), so $k+p$ is typically much smaller than n. We stress that q is a fixed constant with respect to the matrix size rather than a negligible one, with $q = 20$ each power iteration performs two matrix products, so the constant is not small, and the leading-order ratio $n/[q(k+p)]$ should be read with this factor in mind rather than as an unconditional order-of-magnitude gain.

Practical remark. RBPI is dominated by GEMM-type operations (the sketch, the projection, and the two products per power iteration), which are highly optimized on modern GPUs. This is a favorable property, but it does not by itself imply a wall-clock speedup: RBPI also incurs repeated QR factorizations, normalizations, and kernel-launch and synchronization overhead, while dense SVD kernels are themselves highly tuned. The realized wall-clock speedup therefore depends on the matrix shape, the hardware, the implementation, and the number of power iterations, and is best assessed empirically rather than inferred from the flop count alone.

C.2 Approximation Guarantee

We now discuss the sense in which RBPI approximates the leading singular subspace of A. The routine is a standard instance of the randomized subspace-iteration framework of Halko et al. (2011), and its behavior rests on two mechanisms: the range of the sketch tends to align with the dominant singular directions, and each power iteration sharpens this alignment.

Exactness in the noiseless low-rank case. First consider the ideal case in which A has exact rank k, *i.e.,* $\sigma_{k+1} = \cdots = \sigma_r = 0$. Then $A = U_k \Sigma_k V_k^\top$ with $U_k \in \mathbb{R}^{m\times k}$, and the sketch $Y = A\Omega = U_k(\Sigma_k V_k^\top \Omega)$ lies in $\mathrm{range}(U_k)$. As long as $V_k^\top \Omega \in \mathbb{R}^{k\times \ell}$ has full row rank k, which holds almost surely for a Gaussian Ω once $\ell \geq k$, the columns of Y span $\mathrm{range}(U_k)$ exactly. Consequently, the orthonormal factor Q satisfies $QQ^\top A = A$, so $B = Q^\top A$ and its SVD reproduces the true singular triplets, and the lifted factor $U = Q\,U_B$ equals U_k up to sign/rotation. In this regime RBPI is exact, using only $\ell = k + p$ sketch columns rather than operating on all n columns.

Error bound in the general case. When A is only approximately low rank, the projection error is controlled by the trailing singular values. Let Q be the orthonormal basis produced after q power iterations. Corollary 10.10 of Halko et al. (2011), combined with the bound $\sigma_j \leq \sigma_{k+1}$ for $j > k$, gives, in expectation over the Gaussian test matrix Ω with oversampling $p \geq 2$,

$$\mathbb{E}\left\|A - QQ^\top A\right\|_2 \;\leq\; \left[1 + \sqrt{\tfrac{k}{p-1}} + \tfrac{e\sqrt{k+p}}{p}\sqrt{r-k}\right]^{\frac{1}{2q+1}} \sigma_{k+1}. \tag{12}$$

Two features of this bound are worth noting. *(i) Relation to the optimum:* the right-hand side is a multiplicative factor times σ_{k+1}, which by Eckart–Young is exactly the error of the best rank-k approximation, so the projection error is bounded relative to this optimal rank-k spectral error. (The projection $QQ^\top A$ itself has rank at most $\ell = k + p$, so this is a comparison of the rank-ℓ projection error against the best rank-k error, not a claim that RBPI returns the optimal rank-k approximation.) *(ii) Effect of q:* the bracketed factor is raised to the power $1/(2q + 1)$, so as q increases this factor decreases toward 1 and the bound approaches σ_{k+1}, reflecting the increasingly sharp spectral filtering induced by the power scheme. The rate at which the trailing spectrum is suppressed is governed by the spectral separation: because the power scheme maps $\sigma_j \mapsto \sigma_j^{2q+1}$, the relative contribution of trailing components is attenuated at a rate governed by powers of σ_{k+1}/σ_k, so a larger gap yields faster suppression. This is the setting Ares operates in: RLVR gradients are empirically spectrally concentrated (Sec. 1), so the gap below the leading components is typically large and a moderate q can drive the bracketed factor close to 1.

Role of oversampling. The oversampling parameter p enters equation 12 through the terms $\sqrt{k/(p-1)}$ and $\frac{e\sqrt{k+p}}{p}\sqrt{r-k}$, both of which decrease as p increases. Intuitively, drawing $\ell = k + p > k$ random directions makes the sampled block $V_k^\top \Omega$ better conditioned and more likely to capture all k leading directions, providing robustness against a near-degenerate random projection. A small constant oversample, such as the $p = 10$ used in our implementation, adds robustness to the randomized projection while increasing the cost only linearly in $\ell = k + p$, which is absorbed into the $O(q\,mn\ell)$ term above. We state this as an expected-error effect and do not claim a specific failure probability, which would require a separate high-probability analysis.

From subspace approximation to low-rank reconstruction. The projection bound equation 12 translates directly into a reconstruction bound for the compressed representation. Let $B = Q^\top A$ and compute its exact SVD $B = U_B \Sigma V^\top$. Then

$$QB = Q\,U_B\,\Sigma V^\top = QQ^\top A, \tag{13}$$

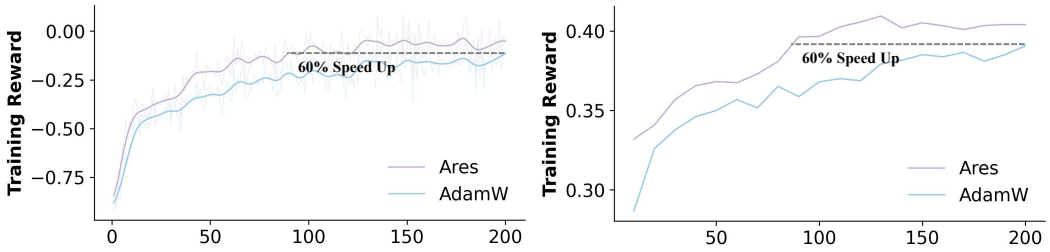


Figure 7: The comparison of training efficiency on Qwen3-4B-Base between ours Ares and AdamW.

and, because Q has orthonormal columns and hence preserves the spectral norm of the residual,

$$\left\|A - Q\,U_B\,\Sigma V^\top\right\|_2 = \left\|A - QQ^\top A\right\|_2. \tag{14}$$

Thus the randomized subspace bound directly controls the reconstruction error of the compressed representation used by Ares. This is a statement about the reconstruction $QU_B\Sigma V^\top$ as a whole; it does not by itself imply that each returned singular vector or singular value matches its true counterpart with the same error, since the accuracy of individual singular vectors additionally depends on the spectral gap through perturbation results of the Davis–Kahan/Wedin type. When the top-k singular values are well separated from the remainder, *i.e.,* the regime RLVR gradients empirically exhibit, such perturbation results further imply accurate recovery of the associated singular vectors and values, which are the quantities Ares uses to form the low-rank update and to estimate the effective rank from the spectral energy entropy.

C.3 Implication for Ares

In summary, RBPI reduces the rank dependence of extracting the leading spectral subspace from n^2 to $q(k+p)\,n$, *i.e.,* from $O(mn^2)$ to $O(q\,mn\ell)$ with $\ell = k+p$. The corresponding leading-order work ratio scales as $n/[q(k+p)]$, which is favorable when $q(k+p) \ll n$, though the constant contributed by q and the practical overheads mean the realized speedup should be verified empirically rather than asserted from the flop count. On the accuracy side, RBPI is exact when A is genuinely rank-k, and in the general case its projection error is bounded by a multiplicative factor times the optimal rank-k error σ_{k+1}, with the factor decreasing toward 1 as the number of power iterations grows and as the spectral gap widens; oversampling adds robustness to the random projection. Because RLVR gradients empirically display a pronounced spectral gap (Sec. 1), a moderate number of power iterations and a small oversample are expected to let RBPI track the leading subspace closely, which is what makes it a suitable and efficient replacement for a dense SVD inside Ares.

D Further Analysis of Ares

D.1 Comparison of Training Efficiency

Fig. 7 compares the training dynamics of Ares and AdamW on Qwen3-4B-Base. Ares attains the same accuracy as AdamW in substantially fewer training steps and continues to improve after AdamW has plateaued, converging to a higher final accuracy. The comparison here is in terms of optimizer steps; since both methods share the same rollout and reward-verification pipeline per step and the added optimizer overhead is modest (Fig. 4), the step-level efficiency translates into a comparable wall-clock advantage, though we do not claim a fully compute-matched speedup. This efficiency gain is a direct consequence of the low-rank geometry identified in Sec. 1: because the energy of RLVR gradients concentrates in a few leading singular components, restricting the gradient signal to the spectral subspace that carries this energy suppresses uninformative directions and yields a more effective update per step. Optimizing within the geometry that RLVR gradients actually exhibit therefore translates into faster and better convergence.

Table 4: Accuracy of Qwen models trained via GRPO with various optimizers on mathematical tasks. "Avg." refers to average accuracy. The best is in bold and the second is underlined.

Optimizer	AIME24	AIME25	AIME26	AMC23	MATH500	Minerva	OlyBench	Avg.
				Qwen3-1.7B-Base				
Ares (Ours)	15.0	10.8	10.0	43.4	61.6	20.4	29.4	27.2
w/o Rank Momentum	10.8	7.5	5.8	41.9	57.9	18.2	26.7	24.1
w/o Dynamic Rank	12.9	9.2	5.0	48.1	61.9	19.7	29.5	26.6
w/o Low Rank	0.0	0.0	0.4	4.4	9.1	2.9	2.0	2.7
w/o Spectral Normalize	10.8	6.7	5.0	48.1	61.9	20.0	28.0	25.8
w/o Adaptive LR	10.0	5.0	4.2	39.4	60.5	17.4	25.9	23.2
AdamW	7.9	4.2	5.0	37.8	59.2	15.5	25.7	22.2
				Qwen3-4B-Base				
Ares (Ours)	25.4	17.5	20.0	65.0	73.4	28.2	40.3	38.5
w/o Rank Momentum	20.8	19.2	17.5	64.1	71.9	27.5	38.7	37.1
w/o Dynamic Rank	23.3	20.0	17.9	62.5	71.6	28.9	38.7	37.6
w/o Low Rank	0.8	1.7	0.8	17.2	23.6	7.6	8.2	8.6
w/o Spectral Normalize	14.2	11.3	10.0	60.0	73.7	26.7	39.6	33.6
w/o Adaptive LR	20.8	18.8	18.8	65.0	73.3	26.6	39.4	37.5
AdamW	21.3	16.3	13.8	60.9	71.9	27.4	39.1	35.8

Table 5: Accuracy of Qwen models trained via GRPO with various optimizers on mathematical tasks. "Avg." refers to average accuracy. The best is in bold and the second is underlined.

Optimizer	AIME24	AIME25	AIME26	AMC23	MATH500	Minerva	OlyBench	Avg.
				Qwen3-1.7B-Base				
Exp	15.0	10.8	10.0	43.4	61.6	20.4	29.4	27.2
Linear	10.8	7.5	5.8	41.9	57.9	18.2	26.6	24.1
Threshold	7.1	3.8	4.6	39.1	59.0	16.8	25.8	22.3
				Qwen3-4B-Base				
Exp	25.4	17.5	20.0	65.0	73.4	28.2	40.3	38.5
Linear	24.6	24.6	17.1	64.4	72.5	28.4	40.2	38.8
Threshold	20.4	15.0	15.0	56.3	71.5	25.9	37.9	34.6

D.2 Detailed Results of Ablation Study

Table 4 extends the ablation of the main text to Qwen3-4B-Base, and the conclusions are consistent across scales. The most striking result is the collapse of *w/o Low Rank*: removing the low-rank projection and updating in the full space drops the average accuracy to 2.7 on Qwen3-1.7B-Base and 8.6 on Qwen3-4B-Base, far below even AdamW. As noted in the main text, this row keeps the other Ares components (scale s, spectral normalization, and learning rate) fixed at values tuned for a low-rank gradient rather than re-tuning them for a full-rank update, so it is a controlled removal rather than an AdamW-equivalent baseline; the collapse indicates that the components are jointly calibrated around the low-rank gradient rather than that any single full-rank optimizer is unusable. Read together with the separately tuned AdamW baseline (22.2/35.8), it confirms the central premise of Sec. 1: RLVR gradients carry their useful signal in a low-dimensional subspace, and confining the update to that subspace is what makes the optimizer work rather than an incidental efficiency trick. The remaining components each contribute a smaller but consistent gain: dynamic rank, spectral normalization, adaptive learning rate, and momentum are all beneficial at both scales, and notably *w/o Dynamic Rank* already trails the full model, indicating that inferring the rank from the gradient spectrum, rather than fixing it, is responsible for part of the improvement.

D.3 Detailed Results of Various Rank Adaptation Methods

Table 5 reports the full results behind Fig. 6, comparing three strategies for translating the estimated spectral energy into an update rank: an exponential mapping (Exp), a linear mapping (Linear), and a hard cutoff (Threshold). The smooth mappings clearly outperform the hard cutoff on both scales:

Table 6: Accuracy of LLMs at different scales trained through different optimizers via GRPO on the MMLU-Pro sub-benchmarks. The best is in bold and the second best is underlined.

Methods	Enginneering	Philosophy	CS	Physics	Math	Economics	Health	Avg.
			Qwen3-1.7B-Base					
AdamW	25.6	29.3	35.0	26.6	27.5	48.9	34.8	<u>32.5</u>
GaLore	27.6	27.5	36.0	25.5	26.6	47.7	35.8	32.4
Ares (Ours)	37.3	28.7	38.7	26.9	27.5	48.9	36.2	**34.9**
			Qwen3-4B-Base					
AdamW	38.7	40.3	54.9	56.7	65.4	65.6	52.6	<u>53.5</u>
GaLore	37.9	41.7	53.2	55.0	64.5	64.3	53.4	52.9
Ares (Ours)	46.9	45.5	58.1	62.4	71.1	69.1	59.2	**58.9**

Methods	Other	History	Chemistry	Biology	Psychology	Law	Business	Avg.
			Qwen3-1.7B-Base					
AdamW	32.5	28.6	25.8	64.5	53.6	21.0	30.3	<u>36.6</u>
GaLore	30.1	30.4	23.1	63.7	53.4	21.4	28.3	35.8
Ares (Ours)	31.9	29.4	26.0	66.3	52.0	21.3	33.6	**37.2**
			Qwen3-4B-Base					
AdamW	49.0	38.3	53.6	73.9	62.0	26.1	61.1	<u>52.0</u>
GaLore	45.3	40.4	52.2	71.5	61.5	26.7	61.7	51.3
Ares (Ours)	49.8	43.8	61.4	77.4	65.8	29.5	65.7	**56.2**

Exp reaches 27.2 / 38.5 and Linear 24.1 / 38.8 average accuracy, whereas Threshold trails at 22.3 / 34.6. The Threshold strategy selects the rank by an abrupt energy cutoff and thus responds discontinuously to the gradient spectrum, discarding components near the boundary and destabilizing the update subspace as the spectrum evolves. In contrast, the smooth mappings let the rank track the concentration of gradient energy continuously over training, which is precisely the behavior motivated in Sec. 1: the effective dimensionality of the update should follow the observed spectrum rather than jump at a fixed threshold.

D.4 Performance on Various Domain Tasks

Although Ares is trained only on mathematical data, we assess whether its advantage transfers beyond mathematics. Table 6 reports accuracy on the fourteen MMLU-Pro sub-benchmarks and Fig. 8 on GPQA. Ares achieves the best average accuracy across the board, improving over both AdamW and GaLore not only on STEM-heavy domains such as engineering, physics, and chemistry but also on distant fields such as philosophy, law, and business, and the margin widens at the 4B scale. The gains on GPQA follow the same pattern. This indicates that the low-rank spectral structure Ares exploits is a general property of RLVR training rather than an artifact of the mathematical reward, so optimizing within the RLVR gradient geometry yields improvements that generalize across domains.

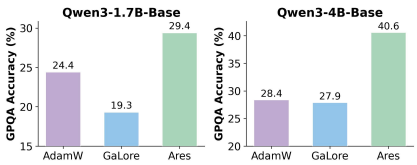


Figure 8: Accuracy of LLMs at different scales trained through different optimizers via GRPO on the GPQA.

D.5 Effectiveness of Momentum

Fig. 9 studies the sensitivity of Ares to the rank momentum coefficient c, which controls the exponential moving average that smooths the per-step effective rank $\hat{r}_t$ into the tracked rank K (Sec. 4.2). Overall, momentum is beneficial and its benefit is stable: it improves performance over the momentum-free variant, consistent with the *w/o Rank Momentum* ablation in Table 4, and it delivers this gain without careful tuning, as accuracy remains consistently strong across a broad range of coefficients rather than depending on a narrowly tuned value.

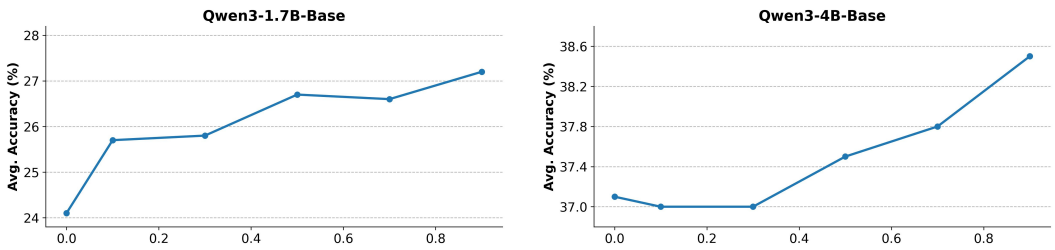


Figure 9: The effectiveness of different momentum coefficients.

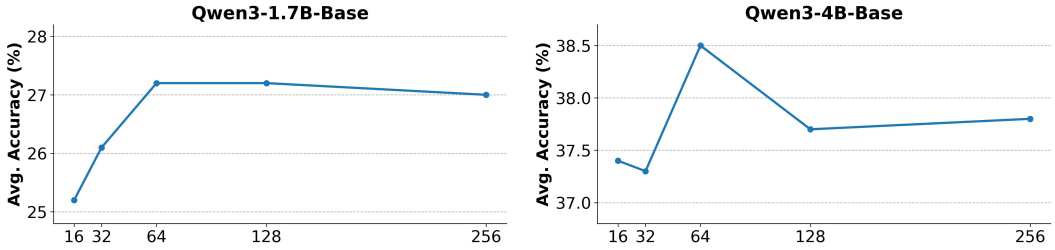


Figure 10: The effectiveness of various initial ranks.

The reason for this behavior lies in how c governs the evolution of the tracked rank K. Because RLVR gradients are estimated from stochastic on-policy rollouts, the per-step estimate $\hat{r}_t$ is noisy, and c determines how much of this noise is passed into K. A larger coefficient smooths $\hat{r}_t$ more strongly, so K evolves gently and stably across steps; the update subspace then follows the slowly drifting spectrum without abrupt jumps, which yields the performance gains observed in the high-c regime. A smaller coefficient, in contrast, lets the per-step noise pass through, so K fluctuates sharply from step to step. These abrupt changes in the update dimensionality destabilize training and degrade performance. Momentum is thus most helpful when it is large enough to keep the rank trajectory smooth, and we adopt a coefficient in this stable regime rather than tuning it precisely.

D.6 Sensitivity Analysis of Initial Rank

Fig. 10 varies the initial rank k, which sets the target-rank budget within which Ares infers and tracks the effective rank (Sec. 4.2). The behavior is governed by how well this budget covers the gradient energy. When k is small, the retained leading subspace cannot capture the vast majority of the spectral energy, so the truncated reconstruction discards non-negligible components and performance is poor. Once k is large enough to cover most of the energy, the picture changes qualitatively: the budget already contains the informative directions, and Ares is free to select the effective rank from within it. In this regime training is stable and accuracy is consistently strong across a wide range of k, in sharp contrast to fixed-rank methods whose accuracy hinges on the chosen r.

This behavior is exactly what the motivation of Sec. 1 predicts: a small rank already suffices to cover the vast majority of the gradient energy (Fig. 1), so once the budget k clears this modest threshold, enlarging it further adds only low-energy directions that Ares adaptively truncates away, leaving performance unchanged. The practical implication is that k needs no careful tuning: one can simply set it somewhat large to safely cover the leading energy, and because the extra directions carry little energy and are discarded online, doing so incurs no meaningful cost. Ares thus removes the brittle rank-tuning burden of static low-rank optimizers, reducing the choice of k from a sensitive hyperparameter to a loose upper bound.

D.7 Adaptation on Various RLVR Algorithms

Ares is defined at the level of the optimizer and is therefore agnostic to the choice of RLVR objective. To verify this, Table 7 replaces AdamW with Ares under three additional algorithms, *i.e.,* RLOO (Ahmadian et al., 2024), REINFORCE++ (Hu, 2025), and GDPO (Liu et al., 2026), beyond

Table 7: Effect of integrating Ares into various RLVR algorithms on mathematical tasks.

	AIME24	AIME25	AIME26	AMC23	MATH500	Minerva	OlyBench	Avg.
				Qwen3-1.7B-Base				
RLOO *w/* AdamW	10.4	5.0	4.6	39.4	60.1	17.6	26.5	23.4
RLOO *w/* Ares	12.5	11.7	8.8	48.8	61.8	19.7	29.6	27.6
Δ	+2.1	+6.7	+4.2	+9.4	+1.7	+2.1	+3.1	+4.2
REINFORCE++ *w/* AdamW	7.5	2.1	2.9	36.7	61.4	17.0	26.4	22.0
REINFORCE++ *w/* Ares	10.0	6.3	4.6	43.4	61.5	19.3	29.0	24.9
Δ	+2.5	+4.2	+1.7	+6.7	+0.1	+2.3	+2.6	+2.9
GDPO *w/* AdamW	6.3	4.6	4.6	40.0	59.3	15.3	25.2	22.2
GDPO *w/* Ares	13.8	8.8	9.6	46.9	62.6	19.3	29.7	27.2
Δ	+7.5	+4.2	+5.0	+6.9	+3.3	+4.0	+4.5	+5.1
				Qwen3-4B-Base				
RLOO *w/* AdamW	19.2	15.8	14.1	64.4	71.1	26.7	38.3	35.7
RLOO *w/* Ares	24.6	20.4	22.5	64.1	73.6	28.6	40.7	39.2
Δ	+5.4	+4.6	+8.4	-0.3	+2.5	+1.9	+2.4	+3.6
REINFORCE++ *w/* AdamW	13.8	11.2	7.9	53.1	71.7	26.1	36.7	31.5
REINFORCE++ *w/* Ares	18.3	15.8	18.3	64.1	73.4	26.7	37.6	36.3
Δ	+4.5	+4.6	+10.4	+11.0	+1.7	+0.6	+0.9	+4.8
GDPO *w/* AdamW	21.6	16.7	16.7	59.7	72.2	26.7	39.2	36.1
GDPO *w/* Ares	22.9	18.8	15.4	60.9	71.4	28.1	40.2	36.8
Δ	+1.3	+2.1	-1.3	+1.2	-0.8	+1.4	+1.0	+0.7

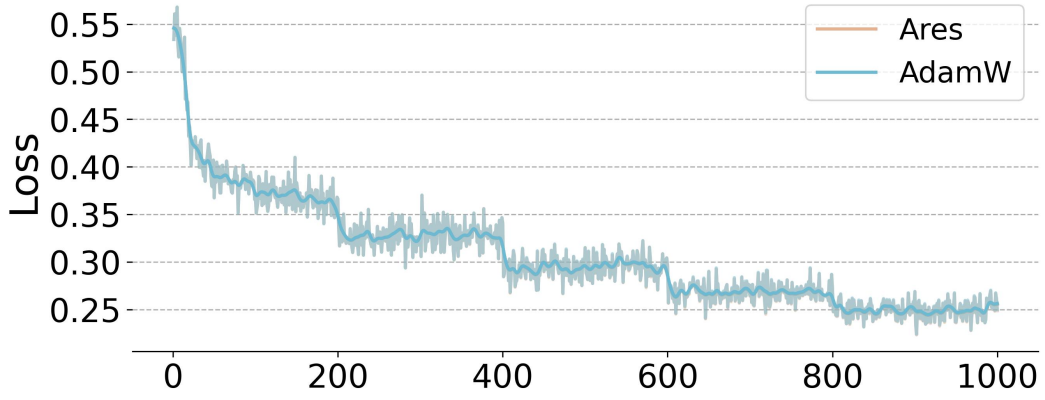


Figure 11: Loss changes during SFT training with different optimizers.

the GRPO setting of the main results. Ares improves the average accuracy for every algorithm at both scales, by up to $+5.1$ points, with only a handful of per-benchmark regressions. The consistency of these gains reinforces the analysis in Sec. 1: the concentration of gradient energy is a property of RLVR training in general, driven by the verifiable reward signal rather than by any particular advantage estimator, so an optimizer that adapts to this shared spectral structure benefits across RLVR algorithms.

D.8 Effectiveness of Ares on SFT Process

Finally, we test whether the advantage of Ares is specific to RLVR by applying it to supervised fine-tuning (SFT). Fig. 11 shows that Ares and AdamW follow nearly identical loss curves, and Table 8 shows that their downstream accuracies are on par (35.3 vs. 34.8 average), with no consistent gain. This is exactly the outcome anticipated in Sec. 1: SFT gradients are far less spectrally concentrated than those of RLVR, so there is little low-rank structure for Ares to exploit and it reduces to a standard adaptive optimizer. The contrast is the strongest evidence for our central thesis, *i.e.,* the

Table 8: Performance of Qwen3-4B-Base through SFT training with different optimizers.

	AIME24	**AIME25**	**AIME26**	**AMC23**	**MATH500**	**Minerva**	**OlyBench**	**Avg.**
AdamW	17.0	21.0	17.3	58.3	69.2	26.5	34.3	34.8
Ares	19.9	20.3	18.1	58.3	69.5	26.7	34.3	35.3

benefit of Ares comes not from the low-rank machinery perse but from matching the optimizer to the distinctive gradient geometry of RLVR. Optimizers should be designed for the geometry of the gradients they actually see.