LINEAR DYNAMICS IN THE RLVR TRAINING OF LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning with verifiable rewards (RLVR) has driven significant performance gains in reasoning-oriented large language models (LLMs), yet its internal training dynamics remain largely a black box. In this work, we perform a comprehensive trajectory-level analysis of RLVR and uncover a striking regularity: across various model families, RL algorithms, and training configurations, RLVR consistently enters a *robust linear regime*, where both parameter weights and output log-probabilities, measured rigorously via teacher-forced evaluation, evolve in a highly linear manner ($R^2 > 0.7$). Through controlled experiments and analysis, we find that this linearity is not a coincidence, and offer a possible explanation based on the high-variance, noisy nature of RLVR training signals, which may act as a low-pass filter to concentrate optimization along a stable, low-dimensional drift. Moreover, we show that this linear dynamic is not merely descriptive but powerfully predictive and actionable. Specifically, weight-space extrapolation matches the performance of standard RL optimization while achieving a $6.1\times$ training speedup through periodic regrounding. Meanwhile, output-space extrapolation serves as a lightweight intervention that effectively bypasses late-stage model collapse, consistently outperforming standard RL across mathematical and coding benchmarks, with an average performance improvement of 4.2%. Our code is available at `https://anonymous.4open.science/r/RLVR-Linearity-ICLR-31CD`.

1 INTRODUCTION

Reinforcement learning with verifiable rewards (RLVR) has emerged as a pivotal stage in the post-training of reasoning-oriented large language models (LLMs), leading to high performance gains in math and coding (OpenAI et al., 2024; Lambert et al., 2025; Guo et al., 2025). However, the black-box nature of LLMs leaves their evolution during this stage largely unknown. A deeper understanding of RLVR training dynamics is thus essential, both to explain where the performance gains come from and to improve the controllability, stability, and efficiency of post-training.

Recent work has examined RLVR dynamics from several complementary perspectives, including exploration-exploitation (Cheng et al., 2025; Agarwal et al., 2025), parameter-space dynamics (Mukherjee et al., 2025; Zhu et al., 2025), and behavioral patterns (Lin et al., 2025; Gandhi et al., 2025). Despite this growing body of work, these studies have largely focused on either too microscopic per-step model updates or too macroscopic behavior changes after the whole training process, leaving the training path itself underexplored. Per-step analyses can characterize individual weight updates, but they lack a global view of the whole training process and the correlation between training steps. For example, a specific pattern reinforced in an RLVR step might be suppressed in subsequent ones, causing their effects to ultimately cancel out or accumulate into a coherent stronger direction in the long run. On the other hand, macroscopic analysis only reveals the high-level changes in model behaviors—such as shifts in response length, or the emergence of self-correction behaviors—without understanding the internal mechanisms. Thus, several central questions in RLVR are left unanswered because of their inherent dependence on trajectory-level analysis, including where performance gains come from, or whether RLVR instills genuinely new reasoning capabilities (Wen et al., 2026; Wang et al., 2026) or primarily elicits latent ones (Wu et al., 2026; Yue et al., 2025).

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.

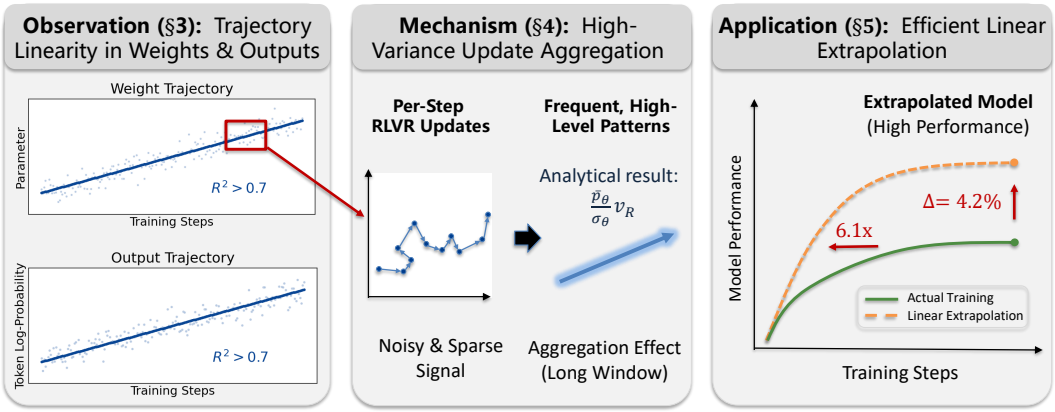

Figure 1: **Overview of our main findings.** In § 3, we show that model weights and outputs follow highly linear trajectories during RLVR, with median R^2 values consistently exceeding 0.7 across diverse settings. In § 4, we link this behavior to the aggregation of noisy, high-variance RL updates into a stable global drift. In § 5, we exploit this structure to predict future model states from early checkpoints, which reduces training cost and improves downstream performance.

In this work, we perform a comprehensive trajectory-level analysis of RLVR dynamics. Our empirical investigation reveals a striking phenomenon: the evolution of both LLM weights and outputs consistently exhibits a high degree of trajectory linearity. We characterize this as a *robust linear regime* (§3), which persists throughout the majority of the training process. Concretely, both weight and output trajectories achieve a coefficient of determination (R^2) exceeding 0.7. This pattern is consistent across a broad range of base models (DeepSeek-R1-Distill-series (Guo et al., 2025), Open-Nemotron-1.5B (Moshkov et al., 2025), Qwen3-series (Yang et al., 2025)), RL algorithms (GRPO (Shao et al., 2024), REINFORCE++ (Hu et al., 2025) and GSPO (Zheng et al., 2025a)), and training setups, suggesting that RLVR trajectories are far more structured than their apparent complexity suggests. The observed linearity implies a surprising degree of directional stability: rather than continually discovering new directions, RLVR enters a regime where later updates increasingly reinforce and amplify directions established earlier. This raises a natural puzzle: *Why does a highly nonlinear Transformer trained with stochastic RL exhibit stable linear dynamics?*

We first provide a mechanistic account of how this linearity emerges (§ 4). We rule out several plausible explanations: e.g., that the small learning rate confines updates to a locally linear neighborhood, or that the AdamW optimizer smooths nonlinear fluctuations. Instead, our analysis points to the high-variance training signal of RLVR as a plausible contributor to this linearity. Sparse and noisy reward signals may suppress update directions led by infrequent patterns, while allowing frequent patterns to accumulate into a stable, low-dimensional drift. This provides a possible explanation for why noisy per-step updates can produce an approximately linear trajectory over longer training windows. Controlled noise-injection experiments further support this interpretation, showing that the observed linearity is not a coincidence but a consequence of structured dynamics intrinsic to RLVR.

We further show that this linear structure is not only descriptive but also predictive (§ 5). Specifically, a future model can be obtained by linearly extrapolating from existing training steps. Experiments on multiple benchmarks (AIME24/25 (Zhang & Team Math-AI, 2024; 2025), MATH500 (Lightman et al., 2024), and LiveCodeBench (Jain et al., 2025)) show that the extrapolated models attain performance very close to that of continued training, even when the extrapolation horizon is large. For output-space extrapolation, the predicted model can even surpass the point of model collapse and outperform actually trained checkpoints. The linear regime is thus mechanistically grounded enough to support accurate forecasting, which incidentally opens up new avenues for efficient post-training.

Our contributions are threefold: (1) We uncover a robust linear regime in RLVR, across model families, RL algorithms, and training setups, with model parameters and outputs consistently exhibiting median R^2 values above 0.7; (2) our analysis and controlled experiments suggest that high-variance RL signals help concentrate optimization along a stable global drift; and (3) we exploit this structure through trajectory extrapolation, achieving up to a $6.1\times$ reduction in gradient-based training and a **4.2%** average improvement through output-space extrapolation.

2 PRELIMINARIES IN RLVR

Reinforcement Learning with Verifiable Rewards (RLVR) bypasses the noisy reward models of standard RLHF by optimizing language models against deterministic, rule-based feedback (e.g., mathematical correctness) (Shao et al., 2024; Le et al., 2022). To mitigate the memory overhead of the actor-critic architecture in standard PPO (Schulman et al., 2017), modern RLVR heavily relies on critic-free algorithms.

A representative algorithm in this paradigm is Group Relative Policy Optimization (GRPO) (Shao et al., 2024). For a policy π_θ and a given prompt x, GRPO samples a group of G responses $\{y_i\}_{i=1}^{G}$ from the reference policy $\pi_{\theta_{\text{old}}}$. The importance weight for the t-th token is defined as $w_{i,t}(\theta) = \frac{\pi_\theta(y_{i,t}|x,y_{i,<t})}{\pi_{\theta_{\text{old}}}(y_{i,t}|x,y_{i,<t})}$. Since $r(x, y_i)$ is a response-level reward, the advantage is uniform across tokens within each response and is computed by standardizing within the group:

$$\hat{A}_{i,t} = \hat{A}_i = \frac{r(x, y_i) - \text{mean}\big(\{r(x, y_i)\}_{i=1}^{G}\big)}{\text{std}\big(\{r(x, y_i)\}_{i=1}^{G}\big)}.$$

The GRPO objective integrates these group-normalized advantages with a PPO-style surrogate loss:

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{x\sim\mathcal{D},\{y_i\}_{i=1}^{G}} \left[\frac{1}{G}\sum_{i=1}^{G} \frac{1}{|y_i|} \sum_{t=1}^{|y_i|} \min\Big(w_{i,t}(\theta)\hat{A}_{i,t},\ \text{clip}\big(w_{i,t}(\theta), 1\pm\epsilon\big)\hat{A}_{i,t} \Big) \right].$$

For simplicity in our trajectory analysis, we focus on the core surrogate objective, omitting the KL divergence penalty typically used to prevent policy drift.

3 EMPIRICAL PHENOMENON: RLVR TRAJECTORIES ARE ALMOST LINEAR

In this section, we quantitatively demonstrate that RLVR trajectories exhibit striking linearity, both in the weight (parameter) space and the output-space. To rule out the possibility that this phenomenon is an artifact of specific experimental configuration, we evaluate a total of 13 diverse settings, showing that this linearity persists robustly across various models, data regimes, and RL algorithms.

3.1 LINEARITY IN WEIGHTS

We first investigate the linearity of model weights during RLVR. We collect intermediate checkpoints from the DeepScaleR pipeline (Luo et al., 2025) (Appendix A) and run a per-weight linear regression on the trainable parameters, using the coefficient of determination R^2 to quantify trajectory linearity.

As shown in Figure 2 (a, left), the distribution of R^2 values is concentrated near 0.8, with over 70% of weights achieving $R^2 > 0.7$. In contrast, the R^2 distribution for SFT on the same base model is centered primarily around 0.4, indicating significantly weaker linearity. This surprising observation indicates that, for most parameters, the full training trajectory is well approximated by a linear trend. Figure 3 (left four panels) shows representative weight trajectories with near-linear increasing and decreasing trends despite local fluctuations.

To further characterize linearity across different layers and components of LLMs, we compute the average R^2 for each layer. As shown in Figure 7 (Appendix B.1), high linearity is consistent across the vast majority of model components, including attention and feed-forward (MLP) layers.

3.2 LINEARITY IN OUTPUT LOGITS AND INTERMEDIATE ACTIVATIONS

We next ask whether the linear evolution of weights translates to the linear updates of model outputs and intermediate activations.

To do so, we sample a fixed set of reasoning traces, generated by the base model, and use them as probes to track the evolution of log probabilities throughout training. Details regarding probe construction and teacher-forced evaluation are provided in Appendix B.2.

As shown in Figure 2 (a, right), the distribution of token-level R^2 values is strongly concentrated near 0.8, indicating that output behavior evolves in a highly linear manner over the course of RLVR

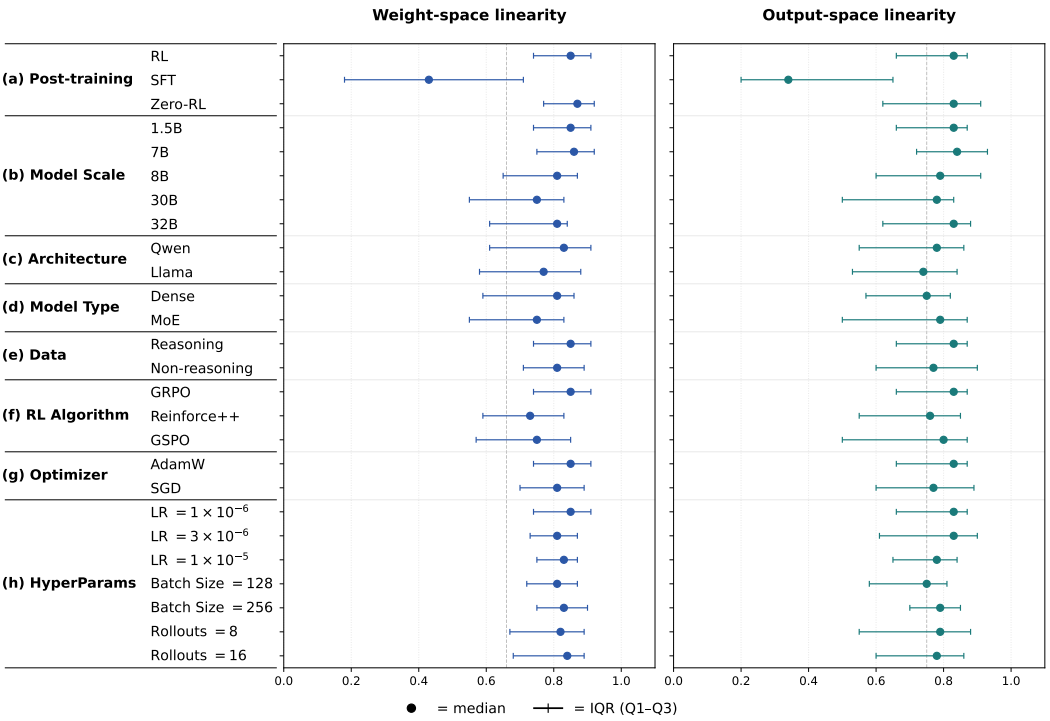

Figure 2: RLVR trajectory linearity across diverse experimental settings. Each row corresponds to a specific configuration grouped by category. **Left:** Weight-space linearity. **Right:** Output-space linearity. Horizontal lines represent the interquartile range (IQR) and dots indicate the median R^2 for each setting. **Vertical dashed lines** at $R^2 = 0.7$ highlight the threshold for strong linearity.

training. Figure 3 (right four panels) presents examples of probing tokens: the log-probabilities of "wait" and "but" show increasing trends across RL steps, whereas those of "so" and "alternatively" decrease. These shifts may reflect changes in how the model structures and revises its reasoning.

We also observe that tokens with larger shifts in log-probabilities tend to display stronger linearity, suggesting that the most substantial changes in model output are often the most structurally organized. A similar conclusion can be drawn on intermediate activations (Figure 4(c)). Due to page constraints, we defer discussion on activation linearity and the fine-grained analysis of different token categories to Appendix B.2. Collectively, these results show that the linear regime of RLVR extends beyond weight space and is clearly reflected in token-level model behavior.

3.3 Is the observed linearity a coincidence?

While we have observed strong linearity in RLVR, a critical question remains: could this phenomenon be a coincidence arising from a specific combination of base model, training data, RL algorithm and hyper-parameters? To address this, we verify the linearity of RLVR trajectories across a wide range of settings. On the model side, we vary model scale (from 1.5B to 32B parameters), architecture (Qwen and Llama), and parameterization type (dense and MoE). On the data side, we consider both reasoning and non-reasoning regimes, including datasets such as DeepScaleR and RLVR-IFeval. On the algorithm side, we evaluate multiple mainstream RL methods, including GRPO, REINFORCE++, and GSPO. Finally, on the training setup side, we experiment with different learning rates, batch sizes and numbers of rollouts.

As shown in Figure 2 (b)-(h), strong linearity persists across all evaluated configurations. In every case, both weight-level and token-level R^2 remain above 0.7, demonstrating robustness to substantial variations in model scale, architecture, base model, training data, RL algorithm, and hyperparameters. Detailed results for each setting are presented in Table 6 (Appendix B.3), with further experimental and training details provided in Appendix A.

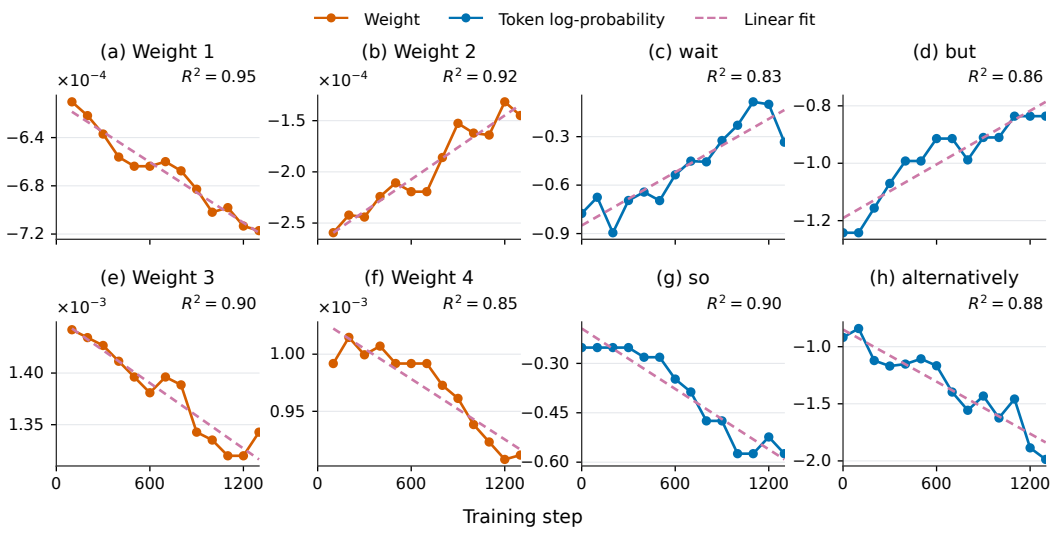

Figure 3: Representative weight and token log-probability trajectories during RLVR. **Left four panels:** Individual weight dimensions (orange). **Right four panels:** Token log-probabilities evaluated under teacher forcing on fixed probe traces (blue). Dashed pink lines show linear fits, with R^2 reported in each panel.

4 A DEEPER LOOK OF THE ORIGINS OF LINEARITY DURING RLVR

Our preceding experiments establish pervasive linearity in model trajectories across various RLVR configurations. Given the inherently nonlinear nature of Transformers, this phenomenon remains highly counterintuitive. In this section, we uncover the core mechanisms underlying such linearity. We first eliminate multiple confounding factors that could account for weight-space linearity, then pinpoint a key origin of the linearity, and finally characterize how linear weight updates propagate to intermediate activations and final model outputs.

4.1 FACTORS THAT DO NOT INDUCE RLVR LINEARITY

We first evaluate three plausible candidate explanations for RLVR's linear training dynamics: small weight update magnitudes, optimizer choice, and SFT initialization before reinforcement learning. Controlled experiments demonstrate that none of these factors serve as the fundamental cause.

Small weight update magnitudes. To stabilize training, RLVR typically employs a small learning rate (e.g., $1e^{-6}$), which results in limited weight modification and could confine optimization to a locally linear region in parameter space. However, we observe that even for weights exhibiting more than 5% relative change, the average R^2 still exceeds 0.7. Furthermore, when increasing the learning rate to $1e^{-5}$, training becomes less stable and performance degrades (e.g., AIME24 accuracy drops roughly 30%), yet the weight trajectories remain highly linear ($R^2 > 0.7$; see Figure 2 (h)). These results indicate that the small magnitude of weight updates is not the primary cause of the linearity.

Optimizer choice. A second hypothesis is that AdamW's momentum smooths noisy updates and reinforces consistent directions. To isolate optimizer effects, we replace AdamW with vanilla SGD while keeping all other configurations fixed (Appendix C). The resulting R^2 distribution remains tightly concentrated above 0.7 (Figure 2 (g)), so the linearity is not driven by AdamW.

SFT initialization prior to RL. A third hypothesis attributes linearity to SFT before RL, which may anchor the model in a flat and locally linear subspace. We rule this out by running RLVR directly on the Qwen3-8B base model with no SFT initialization (denoted as "zero-RL"; see Appendix C). As Figure 2 (a) shows, the weight R^2 still predominantly lie above 0.7, so the linearity persists without any SFT initialization.

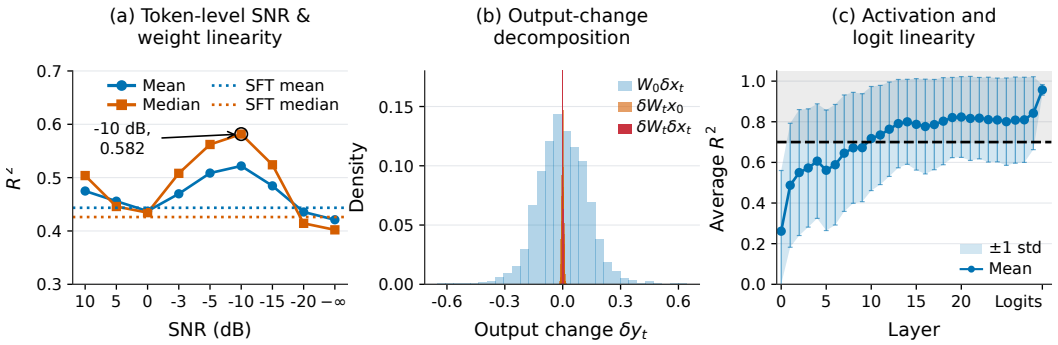

Figure 4: **Mechanisms and propagation of RLVR linearity. (a)** Active-parameter weighted mean and median R^2 under token-level noise. Dotted lines denote vanilla SFT; $-\infty$ denotes zero signal. **(b)** Contributions from input, weight, and second-order changes in a representative MLP layer. **(c)** Layer-wise mean R^2 of activation trajectories. Shading and error bars show ± 1 standard deviation across selected neurons; the final point corresponds to logits. The dashed line marks $R^2 = 0.7$.

4.2 Noisy training signal during RLVR is closely related to its linear dynamics

Beyond the factors above, another key property of RLVR is that its training signal is highly abstract and noisy. Leading RLVR algorithms, such as GRPO and REINFORCE++, assign token-level credit solely based on final answer correctness. This design introduces high variance: locally correct token choices may receive negative rewards when subsequent errors lead to an incorrect final answer, while locally incorrect ones may receive undeserved positive rewards when later reflection steps recover the outcome. In this part, we give an intuitive explanation of why the noisy training signals of RLVR can potentially lead to its linearity. We then establish the empirical relationship between the noise level of the training signal and the linearity of the training trajectories.

Under noisy supervision, models prioritize *frequent, high-level patterns* and act as low-pass filters that suppress sporadic, fine-grained variations (Han et al., 2025; Arpit et al., 2017). This aligns with recent work on credit assignment, which shows that uniformly distributing the final outcome reward across tokens prevents models from learning rare, low-level actions (Parthasarathi et al., 2025; Li et al., 2026). Concretely, the per-step gradient is highly noisy at the token level, but, when averaged, individual sample gradients largely cancel and the expected update concentrates in a low-dimensional subspace spanned by the most frequent successful patterns. Within that subspace, the optimization direction is approximately fixed, which produces a stable linear drift in weight space.

We analyze how reward variance enters the GRPO update in Appendix D and find that its gradient can be bounded from a fixed direction $\frac{\bar{p}_\theta}{\sigma_\theta} v_R$. Both the scale of this reference gradient and the deviation bound contain the reward standard deviation σ_θ in the denominator. When the reward retains sufficient variance, σ_θ remains in a stable, sufficiently large range, so small perturbations during training do not substantially change the gradient scale and the deviation bound is tighter. This interpretation is consistent with our empirical observation near entropy collapse in Appendix G: as the reward variance decreases, the gradient scale becomes more sensitive to small perturbations, and the trajectory correspondingly loses linearity. This provides an intuitive account of why the high-variance regime can support stable update dynamics, while the empirical results in the following paragraphs directly assess the relationship between noise and trajectory linearity.

Empirical verification. To validate this account, we test whether injecting reward-style noise into a normally-deterministic objective increases trajectory linearity. Reducing the noise of RLVR via per-token Monte Carlo Tree Search would be the most direct test but is computationally infeasible because of the explosion in rollouts. We therefore go in the other direction: starting from Supervised Fine-Tuning (SFT), whose gradients are highly deterministic and far less noisy than RLVR, we artificially inject controlled noise into the SFT signal (Appendix C). Token-level and sequence-level noise increase the median R^2 of weights from a baseline of 0.426 to 0.582 and 0.568, respectively. Furthermore, as illustrated in Figure 4(a), R^2 exhibits a non-monotonic trend: it initially rises to a peak as noise increases before eventually declining for excessively large noise. This direct evidence

confirms that gradient noise actively contributes to weight linearity by filtering out fine-grained optimization directions, locking the model into amplifying a robust, low-dimensional drift, exactly as our analysis predicts.

From weight linearity to activation and output linearity. Although a Transformer is nonlinear, its activation and output changes can remain approximately linear when the weight and hidden-state changes are relatively small so that first-order effects dominate. Figure 4(b) illustrates this idea for the down-projection layer of the final MLP. For its output $y_t = W_t x_t$, the change can be decomposed exactly as $\delta y_t = W_0 \delta x_t + \delta W_t x_0 + \delta W_t \delta x_t$. The first two terms capture the separate first-order effects of changing the input and the weights, whereas the last term is their second-order interaction. Empirically, the accumulated input change remains only 7.7% of the original input magnitude on average, and the second-order term stays centered near zero. Consequently, the approximately linear changes in weights and hidden states propagate to deeper activations and ultimately to the output logits. We provide the full residual-layer expansion and discuss attention and other nonlinear components in Appendix E.

5 Predictive Extrapolation of RLVR Trajectories

To establish that the observed linear structure is a fundamental property of RLVR rather than a retrospective artifact, we rigorously test its predictive power. If the linear regime is mechanistically grounded, it should be possible to forecast future model states directly from earlier trajectory segments. In this section, we test the predictive capability in both output space and weight space. To ensure optimal RL performance, we utilize intermediate checkpoints from DeepScaleR (Luo et al., 2025) as our *Standard RL* baseline, upon which all trajectory extrapolations are anchored. Specifically, we check whether we can predict the output logits and weights in a future step, without actual training. We evaluate our extrapolation methods on AIME24/25 (Zhang & Team Math-AI, 2024; 2025), MATH500 (Lightman et al., 2024) and LiveCodeBench (Jain et al., 2025) benchmarks. Full hyperparameter configurations and evaluation details for these runs are deferred to Appendix A.

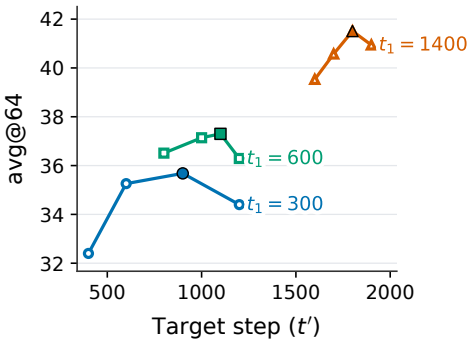

Figure 5: Weight-space extrapolation performance on AIME24 across different second anchor point t_1 and target steps t' for $t_0 = 0$.

5.1 Output-space extrapolation

We start by predicting token logits $l_{t'}$ at a future time step t' based on logits l_{t_0} and l_{t_1} at existing checkpoints at time t_0 and t_1:

$$\mathbf{l}_{t'} = \mathbf{l}_{t_0} + \alpha(\mathbf{l}_{t_1} - \mathbf{l}_{t_0}). \tag{1}$$

Here $\alpha = \frac{t'-t_0}{t_1-t_0} > 1$ is the extrapolation ratio. This formulation provides a simple, training-free mechanism to predict the policy at a future stage. Table 1 compares the extrapolated model with the model from real training on AIME24/25, MATH500, and LiveCodeBench. Across all tasks, the extrapolated model consistently outperforms the standard RL training. This not only confirms the predictive power of output linearity but also offers a path to overcome the instability of late-stage RL training to further boost reasoning performance.

Table 1: Output-space extrapolation ($\alpha = 2$).

Method		Std. RL	Ours	%Imp
AIME24	Avg@64	41.93	**44.58**	+6.3%
AIME25	Avg@64	31.35	**33.33**	+6.3%
MATH500	Avg@64	88.03	**89.00**	+1.1%
LCB	Avg@4	17.86	**19.76**	+10.6%
Macro-average		44.79	**46.67**	+4.2%

5.2 Weight-space extrapolation

A similar extrapolation experiment is conducted in weight space. For a future time step t', we directly predict model weights at t', from two previous checkpoints at time steps $t_0 < t_1$:

$$\mathbf{W}_{t'} = \mathbf{W}_{t_0} + \beta(\mathbf{W}_{t_1} - \mathbf{W}_{t_0}). \tag{2}$$

Table 2: **Performance under Fixed Training Budgets.** We evaluate our proposed approach (extrapolation via periodic re-grounding) against the standard RL baseline. Under fixed gradient computation budgets (actual training steps s), our method consistently yields a higher average performance than the standard optimization path.

Phase	Steps (s)	Method	AIME24 (Avg@64)	AIME25 (Avg@64)	MATH500 (Avg@64)	LCB (Avg@4)	Avg Score	%Imp
Early	200	Standard RL	0.3172	0.2536	0.8421	0.2714	0.4211	—
		Ours	0.3318	0.2979	0.8611	0.2619	0.4382	+5.2%
Mid	400	Standard RL	0.3391	0.2682	0.8525	0.2810	0.4352	—
		Ours	0.3672	0.3005	0.8658	0.2905	0.4560	+6.3%
Late	800	Standard RL	0.3490	0.2932	0.8664	0.2821	0.4477	—
		Ours	0.3984	0.3094	0.8667	0.2905	0.4663	+5.7%
Converged	1200	Standard RL	0.3828	0.2995	0.8658	0.2857	0.4585	—
		Ours	0.4120	0.3120	0.8731	0.2762	0.4683	+2.3%

Here $\beta = \frac{t'-t_0}{t_1-t_0} > 1$ is the weight-space extrapolation ratio. Once the two anchor checkpoints are available, no additional training is required. Unlike output-space extrapolation, this procedure produces a complete model with parameters $\mathbf{W}_{t'}$, ready for further training or deployment.

Figure 5 shows this model's AIME24 performance as the target step t' varies. The curves exhibit an inverted-U pattern: moderate extrapolation improves performance, supporting the predictive validity of the linear direction, whereas projecting too far causes degradation. For anchors at $t_0 = 0$ and $t_1 = 300$, performance peaks around $t' = 900$. The slope estimated over a finite interval $t_1 - t_0$ is imperfect, so larger extrapolation ratios β amplify errors in the predicted weights. Appendix F.1 provides empirical guidelines for choosing t_0, t_1, and β under a given training budget.

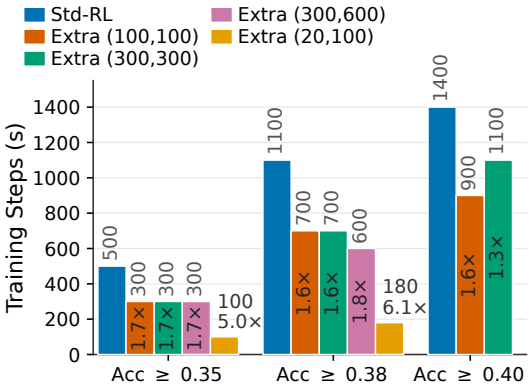

Figure 6: **Effect of periodic re-grounding on RL training efficiency.** Bars are grouped by target AIME24 accuracy levels ($\geq$0.35, $\geq$0.38, $\geq$0.40). Annotated values indicate the speedup factor, defined as the ratio of steps required under standard RL to those with periodic re-grounding.

5.3 Periodic Re-grounding

To mitigate performance degradation from amplified extrapolation errors, we introduce a dynamic approach that interleaves gradient-free weight extrapolation within standard RL optimization. Formally, this process operates in cycles of period $C = m + n$ (configured in Appendix F.2). Each cycle begins with m steps of standard gradient descent to align with the true reward signal, followed by n steps of linear projection to accelerate progress and reduce computational overhead. Let k denote the current global step and η the learning rate. Define $\tau = C\lfloor k/C \rfloor$ as the starting step of the current cycle. The update rule for the model parameters $\mathbf{W}$ using the RL objective function $\mathcal{L}_{\text{RL}}$ is defined as:

$$\mathbf{W}_{k+1} = \begin{cases} \mathbf{W}_k - \eta \nabla \mathcal{L}_{\text{RL}}(\mathbf{W}_k), & \text{if } (k \bmod C) < m, \\ \mathbf{W}_k + \frac{1}{m}\left(\mathbf{W}_{\tau+m} - \mathbf{W}_\tau\right), & \text{otherwise,} \end{cases} \tag{3}$$

Table 2 evaluates periodic re-grounding under matched training budgets. It consistently outperforms standard RL on all three math benchmarks (AIME24, AIME25, and MATH500), while maintaining similar performance on LiveCodeBench.

Figure 6 compares the RL steps required to reach target AIME24 accuracy levels under different re-grounding schedules. For example, to match the standard RL's AIME24 accuracy of 0.40, a

$(100, 100)$ schedule requires only 900 actual RL steps—a $1.6\times$ reduction in gradient-based training. Even a more aggressive $(20, 100)$ schedule reaches > 0.38 AIME24 accuracy with just 180 RL steps, matching the baseline trained for 1100 steps and yielding a $6.1\times$ speedup.

Together, these experiments demonstrate the practical value of linear RLVR trajectories. Output-space extrapolation bypasses late-stage model collapse and improves performance without additional training. Weight-space extrapolation produces a lookahead model that closely matches the performance of further GPU-intensive RL training. Periodic re-grounding then returns extrapolated weights to standard RL optimization, correcting long-horizon inaccuracies and maintaining alignment with the training trajectory to accelerate extended runs.

6 Related Work

Recent studies have increasingly investigated the internal mechanisms of RLVR along two main threads: uncovering the structured parameter dynamics that drive reasoning gains, and exploiting the resulting linear weight space for model merging and extrapolation.

Parameter Dynamics in LLM Post-Training. Despite relying on full-parameter optimizers such as AdamW, recent work shows that RLVR weight updates are in fact tightly constrained along three axes: spatial, geometric, and temporal. Spatially, Mukherjee et al. (2025) observe extreme sparsity in RL fine-tuning, with updates largely confined to small subnetworks (roughly 20% of parameters activated). Geometrically, Zhu et al. (2025) formally prove that RLVR traverses the "path not taken", optimizing primarily along non-principal directions of the Hessian. This orthogonal perturbation explains why RLVR achieves strong reasoning gains with high sample efficiency (Wang et al., 2025) while averting catastrophic forgetting of pre-trained linguistic capabilities. However, both spatial and geometric analyses remain strictly descriptive, offering structural insights without translating them into actionable training procedures. Temporally, this structural rigidity manifests as "Rank-1 Dominance" (Cai et al., 2026), where the parameter update matrix (ΔW) is overwhelmingly governed by its top singular subspace that dictates over 99% of reasoning improvements and exhibits strictly linear dynamics throughout training. Yet, this study merely identifies rank-1 dominance as an empirical phenomenon without providing a corresponding mechanistic explanation.

Moreover, existing studies stop at weight-space artifacts and do not characterize their implications in output space. We address these gaps by mapping low-rank weight dynamics to output log-probabilities, providing a mechanistic account of how low-rank parameter dynamics deterministically propagate to token-level linear evolution and constrain capability acquisition under RLVR.

Model Merging and Extrapolation. A separate line of work exploits the linear weight space for checkpoint combination and extrapolation. Methods such as Model Soup (Wortsman et al., 2022) leverage the linear connectivity of fine-tuned models within low-error basins to perform spatial interpolation, while ExPO (Zheng et al., 2025b) extrapolates between SFT and aligned checkpoints to address cross-stage distribution mismatch. These methods, however, treat linear operability as a useful empirical property rather than asking why it holds. We focus specifically on the RLVR stage and provide a mechanistic explanation for why such simple linear operations on weight space succeed, which in turn enables principled performance prediction along the training trajectory.

7 Conclusion

In this paper, we presented a comprehensive trajectory-level analysis of RLVR. We identified a pervasive *robust linear regime* wherein both model parameters and output space evolve linearly over the course of training. We show that the noisy reward signals in RLVR effectively regularize optimization by filtering out local fluctuations and maintaining a stable macroscopic drift. Leveraging this foundational insight, we introduced practical algorithms for predictive extrapolation in both weight and output spaces. Specifically, weight-space extrapolation, integrated with a periodic re-grounding, can match RL performance with lower overhead, while output-space extrapolation provides a lightweight intervention that prevents late-stage model collapse, allowing models to consistently outperform standard RL. We provide a deeper discussion of the broader implications of our findings in Appendix G. We also discuss the potential limitations of this work in Appendix H.

AI USE STATEMENT

Generative AI tools were used to assist with language polishing and LaTeX formatting. The authors reviewed the resulting edits and take full responsibility for the accuracy and integrity of the final manuscript.

REFERENCES

Shivam Agarwal, Zimin Zhang, Lifan Yuan, Jiawei Han, and Hao Peng. The unreasonable effectiveness of entropy minimization in llm reasoning. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, pp. 107150–107180. Curran Associates, Inc., 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/9a14cf2ee36cf2524f54d8f196a6734c-Paper-Conference.pdf.

Devansh Arpit, Stanisław Jastrzębski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S. Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, and Simon Lacoste-Julien. A closer look at memorization in deep networks. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 233–242. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/arpit17a.html.

Yuchen Cai, Ding Cao, Xin Xu, Zijun Yao, Yuqing Huang, Benyi Zhang, Zhenyu Tan, Guiquan Liu, and Junfeng Fang. On predictability of reinforcement learning dynamics for large language models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=SdHmA6BYVJ.

Daixuan Cheng, Shaohan Huang, Xuekai Zhu, Bo Dai, Wayne Xin Zhao, Zhenliang Zhang, and Furu Wei. Reasoning with exploration: An entropy perspective, 2025. URL https://arxiv.org/abs/2506.14758.

Lénaïc Chizat, Edouard Oyallon, and Francis Bach. On lazy training in differentiable programming. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/ae614c557843b1df326cb29c57225459-Paper.pdf.

Kanishk Gandhi, Ayush K Chakravarthy, Anikait Singh, Nathan Lile, and Noah Goodman. Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective STars. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=QGJ9ttXLTy.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, September 2025. doi: 10.1038/s41586-025-09422-z. URL http://dx.doi.org/10.1038/s41586-025-09422-z.

Andi Han, Wei Huang, Zhanpeng Zhou, Gang Niu, Wuyang Chen, Junchi Yan, Akiko Takeda, and Taiji Suzuki. On the role of label noise in the feature learning process, 2025. URL https://arxiv.org/abs/2505.18909.

Bingxiang He, Zekai Qu, Zeyuan Liu, Yinghao Chen, Yuxin Zuo, Cheng Qian, Kaiyan Zhang, Weize Chen, Chaojun Xiao, Ganqu Cui, Ning Ding, and Zhiyuan Liu. Justrl: Scaling a 1.5b llm with a simple rl recipe, 2025a. URL https://arxiv.org/abs/2512.16649.

Jujie He, Jiacai Liu, Chris Yuhao Liu, Rui Yan, Chaojie Wang, Peng Cheng, Xiaoyu Zhang, Fuxiang Zhang, Jiacheng Xu, Wei Shen, Siyuan Li, Liang Zeng, Tianwen Wei, Cheng Cheng, Bo An, Yang Liu, and Yahui Zhou. Skywork open reasoner 1 technical report, 2025b. URL https://arxiv.org/abs/2505.22312.

Jian Hu, Jason Klein Liu, Haotian Xu, and Wei Shen. Reinforce++: Stabilizing critic-free policy optimization with global advantage normalization, 2025. URL https://arxiv.org/abs/2501.03262.

Arthur Jacot, Franck Gabriel, and Clement Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/5a4be1fa34e62bb8a6ec6b91d2462f5a-Paper.pdf.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=chfJJYC3iL.

Aayush Karan and Yilun Du. Reasoning with sampling: Your base model is smarter than you think. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=Vsgq2ldr4K.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, et al. Tulu 3: Pushing frontiers in open language model post-training. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=i1uGbfHHpH.

Hung Le, Yue Wang, Akhilesh Deepak Gotmare, Silvio Savarese, and Steven Chu Hong Hoi. Coderl: Mastering code generation through pretrained models and deep reinforcement learning. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 21314–21328. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/8636419dea1aa9fbd25fc4248e702da4-Paper-Conference.pdf.

Ziheng Li, Liu Kang, Feng Xiao, Luxi Xing, Qingyi Si, Zhuoran Li, Weikang Gong, Deqing Yang, Yanghua Xiao, and Hongcheng Guo. Outcome-grounded advantage reshaping for fine-grained credit assignment in mathematical reasoning, 2026. URL https://arxiv.org/abs/2601.07408.

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 B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/aca97732e30bcf1303bc22ac3924fd16-Paper-Conference.pdf.

Zicheng Lin, Tian Liang, Jiahao Xu, Qiuzhi Liu, Xing Wang, Ruilin Luo, Chufan Shi, Siheng Li, Yujiu Yang, and Zhaopeng Tu. Critical tokens matter: Token-level contrastive estimation enhances LLM's reasoning capability. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 37906–37918. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/lin25j.html.

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Tianjun Zhang, Erran Li, Raluca Ada Popa, and Ion Stoica. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl. https://pretty-radio-b75.notion.site/DeepScaleR-Surpassing-O1-Preview-with-a-1-5B-Model-by-Scaling-RL-19681902c1468005b 2025. Notion Blog.

Sadhika Malladi, Alexander Wettig, Dingli Yu, Danqi Chen, and Sanjeev Arora. A kernel-based view of language model fine-tuning. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 23610–23641. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/malladi23a.html.

Ivan Moshkov, Darragh Hanley, Ivan Sorokin, Shubham Toshniwal, Christof Henkel, Benedikt Schifferer, Wei Du, and Igor Gitman. Aimo-2 winning solution: Building state-of-the-art mathematical reasoning models with openmathreasoning dataset, 2025. URL https://arxiv.org/abs/2504.16891.

Sagnik Mukherjee, Lifan Yuan, Dilek Hakkani-Tur, and Hao Peng. Reinforcement learning finetunes small subnetworks in large language models. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, pp. 132119–132138. Curran Associates, Inc., 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/bf235a1d6780afd979f2f81676f43413-Paper-Conference.pdf.

Sagnik Mukherjee, Lifan Yuan, Pavan Jayasinha, Dilek Hakkani-Tür, and Hao Peng. Do we need adam? surprisingly strong and sparse reinforcement learning with sgd in llms, 2026. URL https://arxiv.org/abs/2602.07729.

OpenAI, Aaron Jaech, Adam Kalai, et al. Openai o1 system card, 2024. URL https://arxiv.org/abs/2412.16720.

Prasanna Parthasarathi, Mathieu Reymond, Boxing Chen, Yufei Cui, and Sarath Chandar. Grpo-λ: Credit assignment improves llm reasoning, 2025. URL https://arxiv.org/abs/2510.00194.

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

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

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.

Yiping Wang, Qing Yang, Zhiyuan Zeng, Liliang Ren, Liyuan Liu, Baolin Peng, Hao Cheng, Xuehai He, Kuan Wang, Jianfeng Gao, Weizhu Chen, Shuohang Wang, Simon Shaolei Du, and yelong shen. Reinforcement learning for reasoning in large language models with one training example. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=IBrRNLr6JA.

Zhilin Wang, Yafu Li, Shunkai Zhang, Zhi Wang, Haoran Zhang, Xiaoye Qu, and Yu Cheng. New skills or sharper primitives? a probabilistic perspective on the emergence of reasoning in rlvr, 2026. URL https://arxiv.org/abs/2602.08281.

Xumeng Wen, Zihan Liu, Shun Zheng, Shengyu Ye, Zhirong Wu, Yang Wang, Zhijian Xu, Xiao Liang, Junjie Li, Ziming Miao, Jiang Bian, and Mao Yang. Reinforcement learning with verifiable rewards implicitly incentivizes correct reasoning in base LLMs. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=jGbRWwIidy.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 23965–23998. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/wortsman22a.html.

Fang Wu, Weihao Xuan, Ximing Lu, Mingjie Liu, Yi Dong, Zaid Harchaoui, and Yejin Choi. The invisible leash: Why rlvr may or may not escape its origin, 2026. URL https://arxiv.org/abs/2507.14843.

An Yang, Anfeng Li, Baosong Yang, et al. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Qiying Yu, Zheng Zhang, Ruofei Zhu, et al. Dapo: An open-source llm reinforcement learning system at scale. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, pp. 113222–113244. Curran Associates, Inc., 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/a4277440d50f1f15d2cb4c14f7e0c0d2-Paper-Conference.pdf.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, pp. 57654–57689. Curran Associates, Inc., 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/537d5aa768c2d534016a4d06f87bc8fb-Paper-Conference.pdf.

Yifan Zhang and Team Math-AI. American invitational mathematics examination (aime) 2024, 2024.

Yifan Zhang and Team Math-AI. American invitational mathematics examination (aime) 2025, 2025.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. Group sequence policy optimization, 2025a. URL https://arxiv.org/abs/2507.18071.

Chujie Zheng, Ziqi Wang, Heng Ji, Minlie Huang, and Nanyun Peng. Model extrapolation expedites alignment. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1025–1041, Vienna, Austria, July 2025b. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.51. URL https://aclanthology.org/2025.acl-long.51/.

Hanqing Zhu, Zhenyu Zhang, Hanxian Huang, DiJia Su, Zechun Liu, Jiawei Zhao, Igor Fedorov, Hamed Pirsiavash, Zhizhou Sha, Jinwon Lee, David Z. Pan, Zhangyang Wang, Yuandong Tian, and Kai Sheng Tai. The path not taken: RLVR provably learns off the principals. In *NeurIPS 2025 Workshop on Efficient Reasoning*, 2025. URL https://openreview.net/forum?id=N75EWQQnb3.

A EXPERIMENTAL SETUP

We follow experimental setup in DAPO (Yu et al., 2025), including the benchmarks and evaluation setups.

A.1 MODELS

In our empirical evaluation, we utilize a diverse set of base models to ensure the broad applicability of our findings. The evaluated models vary significantly across three main dimensions:

- **Varying Parameter Scales**: Our experiments cover models from lightweight sizes (1.5B) to larger scales (up to 30B total parameters), allowing us to analyze the scaling behavior of our observation.
- **Diverse Architectures**: To ensure our conclusions are not architecture-specific, we select models originating from different foundational families, including the Qwen/Qwen3 series and the LLaMA architecture. We also include NVIDIA Nemotron reasoning models built on top of Qwen backbones to test whether the observed phenomenon persists under different post-training pipelines and training recipes.
- **Dense vs. Mixture-of-Experts (MoE)**: While the majority of our evaluated models are dense networks, we specifically include Qwen3-30B-A3B—an MoE architecture with 30 billion total parameters and approximately 3 billion active parameters per token—to validate our observation on sparse models.

Table 3: **Base models evaluated in our study.** The table outlines the base models used to evaluate RLVR trajectory linearity. For MoE models (e.g., Qwen3-30B-A3B), we report both the total parameter count and the number of active parameters activated during inference.

Model	Architecture	Params (Total / Active)	Type
`DeepSeek-R1-Distill-Qwen-1.5B`	Qwen	1.5B / 1.5B	Dense
`Open-Nemotron-1.5B`	Nemotron	1.5B / 1.5B	Dense
`DeepSeek-R1-Distill-Qwen-7B`	Qwen	7B / 7B	Dense
`DeepSeek-R1-Distill-Llama-8B`	LLaMA	8B / 8B	Dense
`Qwen3-8B`	Qwen3	8B / 8B	Dense
`Qwen2.5-32B`	Qwen	32B / 32B	Dense
`Qwen3-30B-A3B`	Qwen3	30B / $\sim$3B	MoE

A.2 RL ALGORITHMS

To ensure that the generalizability of our observations is not an artifact of a specific optimization objective or training dynamic, we conduct comprehensive experiments across a diverse spectrum of RL algorithms. The evaluated frameworks include:

- **GRPO** (Shao et al., 2024): Representing the state-of-the-art in reasoning-focused LLM alignment, GRPO optimizes the policy by normalizing rewards across a sampled group of generated outputs, enabling effective updates without relying on a value model.
- **REINFORCE++** (Hu et al., 2025): This framework proposes Global Advantage Normalization to replace the local group normalization used in GRPO. It effectively corrects the bias introduced by per-prompt normalization in existing critic-free approaches while maintaining computational efficiency by eliminating the critic model.
- **GSPO** (Zheng et al., 2025a): Designed specifically to enhance the training stability of Mixture-of-Experts (MoE) models, GSPO elevates the optimization granularity from the token level to the sequence level, ensuring more robust convergence.

Table 4: **Hyperparameter settings.** These settings are applied consistently across GRPO, REINFORCE++, and GSPO.

Hyperparameter	Value
KL Loss	No
Entropy Regularization	No
Global Batch Size	[128, 256]
PPO Mini-batch Size	64
Max Response Length	16K
Learning Rate	1×10^{-6} (Constant)
Clip Ratio Range	$[0.8, 1.28]$
Temperature	1.0
Rollout (N)	[5, 8, 16]

Table 5: **Computational resources.** Costs are approximate GPU-hours. "Cost basis" gives the per-run or per-group accounting used to estimate the aggregate compute for the reported experiments.

Experiment group	Scope	Cost basis	Total cost
Post-Training runs	All runs reported in Table 6	$\sim$2,000/run $\times$ 13	$\sim$26,000
Weight-space extrapolation	Re-grounding RL schedules reported in Table 2	$\sim$700/run $\times$ 4	$\sim$2,800

A.3 Training details

The training hyperparameters are shown in the Table 4. We adapt our training codebase from VERL and follow the training recipe of three different RL algorithms, including GRPO, GSPO, and REINFORCE++.

Specifically, we collect the intermediate checkpoints from DeepScaleR (Luo et al., 2025), JustRL (He et al., 2025a) and Skywork-OR1-7B (He et al., 2025b) to ensure optimal performance.

A.4 Computational resources

Table 5 summarizes the computational resources used for the main training experiment groups. Unless otherwise specified, each GPU run used 8 NVIDIA H200 GPUs with 141GB memory per GPU. We report approximate GPU-hours because wall-clock time varied with cluster scheduling and evaluation batch composition. We report only the compute incurred by our post-training and re-grounding runs.

A.5 Evaluation benchmarks

To rigorously assess our proposed extrapolation techniques, we conducted an evaluation across a diverse suite of benchmarks. Importantly, the primary objective of this extrapolation is to verify that the linearity of the model's performance scaling is predictable, and to demonstrate that this predictability can be effectively leveraged to accelerate RLVR training. The specific focus and rationale for each benchmark category are outlined below:

- **Math Reasoning**: AIME24/25 (Zhang & Team Math-AI, 2024; 2025) consist of competition-level mathematics problems from the American Invitational Mathematics Examination, which require advanced, multi-step mathematical problem-solving; and MATH500 (Lightman et al., 2024) is a curated, representative subset of 500 problems from the MATH dataset, spanning various mathematical disciplines and difficulty levels.
- **Code Generation**: LiveCodeBench (v5, Oct 2024 – Feb 2025) (Jain et al., 2025) is a continuously updated benchmark for code generation based on real-world competitive programming platforms (e.g., LeetCode, Codeforces).

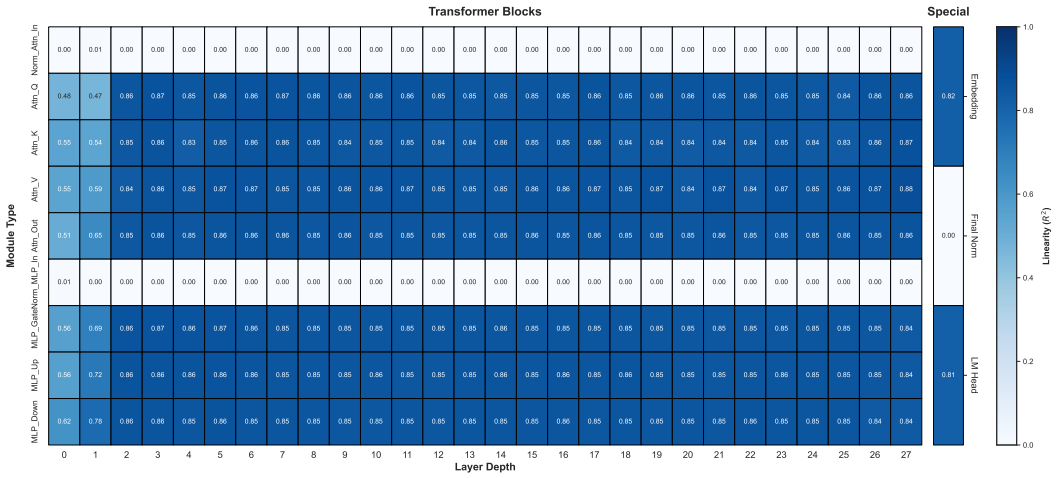

Figure 7: **Layer-wise linearity heatmap of model parameters.** Each cell reports the mean R^2 for a specific module type at a given Transformer layer. Rows correspond to module types, columns correspond to layer indices, and color intensity indicates the magnitude of R^2. The separate "Special" column summarizes non-block components, including the embedding layer, the final normalization layer, and the language modeling head.

A.6 Evaluation setup

We evaluate models on four standard mathematical and code reasoning benchmarks commonly used for assessing reasoning capabilities: AIME'24, AIME'25, MATH500, and LiveCodeBench. All evaluations are conducted in a zero-shot setting. For each question, the maximum generation length is set to $32,768$ tokens under a temperature of 0.6, a top-p value of 0.95.

We report Avg@k and Pass@k, defined as follows: Pass@k measures the proportion of problems where at least one correct solution exists among the top-k samples, reflecting the model's potential coverage. Avg@k denotes the average accuracy (expected Pass@1) calculated over the k samples, reflecting the model's stability.

B Additional Results on Empirical Phenomenon

B.1 Linearity in weights

When calculating the R^2 to quantify the linearity of weights, we exclude weights that remain constant throughout training, since such trajectories are uninformative about update dynamics.

Representative weight trajectories are shown in Figure 3 (left four panels) in the main text.

As illustrated in Figure 7, we analyze the linearity of weight trajectories by calculating the average R^2 for each layer. A notable observation is that all Layer Normalization (LayerNorm) layers exhibit consistently low linearity compared to other layers.

This phenomenon can be attributed to the functional decoupling within the Transformer architecture: LayerNorm parameters primarily govern the statistical normalization of activations, which tends to stabilize early in the training process. In contrast, weights associated with feature transformation (e.g., projection and feed-forward matrices) undergo continuous, linear updates to refine the model's representational capacity.

B.2 Linearity in outputs

Details of Output-Space Linearity Measurement To evaluate output-space linearity, we select a set of queries from AIME24 and use the base model to generate solution trajectories. Specifically, we sample 64 traces per query using the same generation hyperparameters as in our standard evaluation

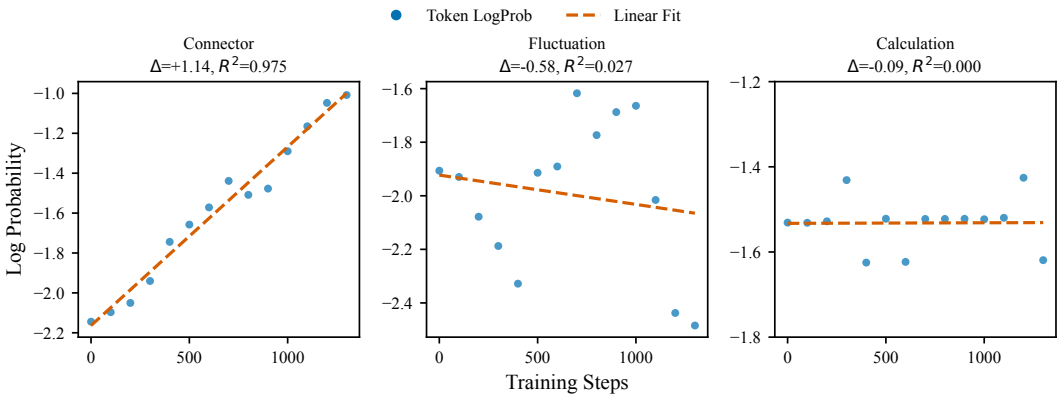

Figure 8: **Case study of token log-probability dynamics. Left:** Tokens acting as logical connectors, characterized by significant log-probability changes and high R^2 values. **Middle:** Tokens with large variations in log-probability but low R^2, where the probability fluctuates irregularly. **Right:** Tokens with smaller log-probability variations, which are mostly components of mathematical calculations.

(Appendix A.6). These trajectories are held fixed throughout the analysis to serve as ground-truth sequences.

For each checkpoint, we compute the conditional log-probability of every generated token using teacher-forcing, i.e., given its preceding context within the same fixed probe trajectory. This yields, for each token position, a trajectory of log-probabilities over training steps. We then fit a linear regression model to each such trajectory as a function of training steps and compute the corresponding coefficient of determination, R^2.

Using fixed probe trajectories under teacher forcing is important for identification: it isolates changes in the model's preference over the same reasoning traces, without conflating them with changes in the sampled outputs themselves. Thus, the resulting R^2 values reflect how linearly the model's token-level preferences evolve during RLVR.

Token categories by magnitude of change and linearity. Beyond the overall R^2 distribution, we further analyze how output linearity varies across different types of token trajectories. We observe a positive association between the magnitude of log-probability change and linearity: tokens with larger behavioral shifts tend to achieve higher R^2. To characterize this pattern more concretely, we group tokens according to their trajectory variance and linearity, as shown in Figure 8.

- **High variance, high R^2.** This is the dominant category. It contains many discourse-level reasoning markers, such as "wait", "but", and "therefore", together with nearby continuation tokens. A representative example is the connector phrase "Wait, that seems a bit messy," whose probability increases substantially and does so in a highly linear manner. This indicates that major behavioral shifts during RLVR are often strongly organized.
- **High variance, low R^2.** A smaller subset of tokens undergoes substantial change without following a clear linear trend. For example, the continuation "Similarly, the two circles of radius 1 are tangent to each other and to AB and BC," exhibits large fluctuations rather than a monotonic trajectory, reflecting more irregular deviations around the dominant linear pattern.
- **Low variance.** Many tokens change little throughout training and therefore form a stable category. A substantial fraction of these tokens are associated with mathematical calculation content, such as "$250 \div 188 = 1 * 188 = 188$, remainder 62," whose log-probabilities remain relatively unchanged during RLVR.

Overall, this categorization suggests that output linearity is not uniform across all tokens: it is strongest precisely for those tokens that account for the largest behavioral shifts.

Table 6: Comprehensive overview of RL training configurations and evaluation results. Each row details a distinct experiment with varying models, training datasets, algorithms, and hyperparameters. Weight R^2 and Token R^2 are computed from model weights and token log-probabilities, respectively; for each metric, we report the median R^2 across training checkpoints and $\Pr(R^2{>}0.7)$, the proportion of checkpoints exceeding the 0.7 threshold. The final row reports an SFT baseline for reference; rollouts are not applicable (—) in this setting.

Model	Data	Algorithm	Optimizer	LR	Batch	Rollouts	Weight R^2		Token R^2	
							Median	$\Pr(>0.7)$	Median	$\Pr(>0.7)$
DS-R1-Qwen-1.5B	DeepScaleR	GRPO	AdamW	1e-6	128	16	0.845	0.794	0.831	0.720
		GRPO	AdamW	3e-6	128	16	0.812	0.739	0.832	0.701
		GRPO	AdamW	1e-5	128	16	0.835	0.762	0.783	0.660
		REINFORCE++	AdamW	1e-6	128	16	0.732	0.707	0.758	0.621
		GSPO	AdamW	1e-6	128	16	0.752	0.717	0.802	0.691
		GRPO	SGD	1e-6	128	16	0.810	0.734	0.768	0.631
Nemotron-1.5B	DeepScaleR	GRPO	AdamW	1e-6	256	8	0.757	0.725	0.732	0.605
DS-R1-Qwen-7B	Skywork	GRPO	AdamW	1e-6	256	16	0.862	0.816	0.835	0.735
DS-R1-Llama-8B	Skywork	GRPO	AdamW	1e-6	256	16	0.774	0.742	0.743	0.694
Qwen3-8B	DAPO-Math	GRPO	AdamW	1e-6	256	8	0.868	0.824	0.829	0.711
Qwen2.5-32B	RLVR-IFeval	GRPO	AdamW	1e-6	128	16	0.811	0.777	0.826	0.705
Qwen3-30B-A3B	Skywork	GSPO	AdamW	1e-6	512	16	0.750	0.702	0.781	0.637
Qwen2.5-1.5B	GSM8K	SFT	AdamW	1e-3	256	—	0.426	0.259	0.344	0.210

Output-Space Quantities To verify if temporal regularity extends beyond token log-probabilities, we analyze the trajectories of logits and intermediate activations across training checkpoints. For each layer, we compute the R^2 between training step and each selected neuron's activation value, then summarize the mean and standard deviation across selected neurons.

As shown in Figure 4(c), mean R^2 generally increases from the initial layers to deeper representations, reaching approximately 0.96 for logits. The initial activation layer has a mean R^2 of approximately 0.26, while many deeper layers are near 0.8; the error bars indicate substantial variation across neurons.

These observations extend the evidence for linearity beyond token log-probabilities to logits and deeper internal representations, while showing that its strength varies across layers and neurons.

B.3 Robustness of linearity

Table 6 presents a comprehensive overview of the training configurations and corresponding linearity evaluations. To ensure the optimal RL performance, we utilize intermediate checkpoints from several established projects: DeepScaleR (Luo et al., 2025) for `DeepSeek-R1-Distill-Qwen-1.5B`, JustRL (He et al., 2025a) for `Open-nemotron-1.5B`, and Skywork-OR1-7B (He et al., 2025b) for `DeepSeek-R1-Distill-Qwen-7B`. Furthermore, the instruction-following RLVR dataset is derived from AllenAI's RLVR-IFeval (Lambert et al., 2025), which samples prompts from the Tulu 2 SFT mixture and augments them with verifiable constraints based on the IFEval taxonomy.

C Additional Results on Origins of Linearity

C.1 Investigation on optimizers

Following Mukherjee et al. (2026), we directly replace AdamW with SGD, setting the learning rate to $[1e\text{-}1, 1e0]$, momentum to 0, and weight decay to 0.

C.2 Investigation on zero-RL

We used the GRPO algorithm to train directly on the base model Qwen3-8B (Yang et al., 2025) using DAPO-Math-17k (Yu et al., 2025).

C.3 Implementation details of noise injection in SFT

Noise-injected SFT objectives. Let

$$\ell_t = -\log p_\theta(x_{t+1} \mid x_{\le t})$$

denote the per-token next-token negative log-likelihood, and let $\tilde{m}_t \in \{0, 1\}$ be the left-shifted SFT mask indicating whether position t predicts a supervised assistant token. The standard SFT loss is

$$\mathcal{L}_{\mathrm{SFT}} = \frac{1}{N} \sum_t \tilde{m}_t \, \ell_t, \qquad N = \sum_t \tilde{m}_t.$$

For **token-level noise injection**, we sample an independent sign for each token,

$$s_t \in \{+1, -1\}, \qquad P(s_t = -1) = p,$$

and define

$$\mathcal{L}_{\mathrm{token}} = \frac{1}{N} \sum_t \tilde{m}_t \, s_t \, \ell_t.$$

For **sequence-level noise injection**, we sample one cut point c and one sign s per sequence,

$$c \sim \mathrm{Uniform}\{0, 1, \ldots, T\}, \qquad s \in \{+1, -1\}, \quad P(s = -1) = p,$$

and retain only the prefix before c:

$$q_t = \mathbf{1}[t < c].$$

The loss is

$$\mathcal{L}_{\mathrm{seq}} = \frac{1}{N} \sum_t \tilde{m}_t \, q_t \, s \, \ell_t.$$

In both variants, the denominator N is unchanged from standard SFT, i.e., we always normalize by the original number of valid supervised tokens rather than by the number of retained noisy tokens.

D Gradient Analysis of GRPO

Let $x \sim \mathcal{D}$ denote the input prompt and $\tau = (y_1, \ldots, y_L)$ be the generated response. The policy $\pi_\theta(\tau \mid x)$ defines a probability distribution over the response space $\mathcal{T}$, and we write $\varphi_\theta(x, \tau) := \nabla_\theta \log \pi_\theta(\tau \mid x)$ for the score function. We consider a binary reward function $R(x, \tau) \in \{0, 1\}$ as a representative case. GRPO normalizes the reward using its mean $\bar{p}_\theta = \mathbb{E}_{x \sim \mathcal{D}, \tau \sim \pi_\theta}[R(x, \tau)]$ and standard deviation $\sigma_\theta = \sqrt{\bar{p}_\theta(1 - \bar{p}_\theta)}$. Each sampled response therefore contributes its score weighted by the normalized reward, and the population ascent direction is

$$g_{\mathrm{GRPO}}(\theta) = \mathbb{E}_{x \sim \mathcal{D}, \tau \sim \pi_\theta} \left[\left(\frac{R(x, \tau) - \bar{p}_\theta}{\sigma_\theta} \right) \varphi_\theta(x, \tau) \right]. \tag{4}$$

This is the standard normalized policy gradient: the baseline $\bar{p}_\theta$ and the scale σ_θ enter as stop-gradient statistics, so Eq. (4) is the update field we analyze rather than the derivative of an objective in which these statistics also depend on θ. To focus on the core optimization dynamics, we first omit the old-policy importance ratio, clipping, and finite-group normalization, and we treat each of them explicitly at the end of this section.

For the remainder of this analysis, we omit x for brevity. Let θ_0 be the initial parameters and $\delta = \theta - \theta_0$. We write $S = \{\tau : R(\tau) = 1\}$ for the success set, $\bar{p}_0 = \Pr_{\theta_0}(S)$ for its initial probability, and

$$v_R = \mathbb{E}_{\pi_{\theta_0}} [\varphi_{\theta_0}(\tau) \mid \tau \in S] \tag{5}$$

for the average score of successful responses at initialization.

We first decompose $g_{\mathrm{GRPO}}(\theta)$ into the score at θ_0 and a drift term:

$$g_{\mathrm{GRPO}}(\theta) = \mathbb{E}_{\pi_\theta} \left[\frac{R(\tau) - \bar{p}_\theta}{\sigma_\theta} \varphi_{\theta_0}(\tau) \right] + \mathbb{E}_{\pi_\theta} \left[\frac{R(\tau) - \bar{p}_\theta}{\sigma_\theta} (\varphi_\theta(\tau) - \varphi_{\theta_0}(\tau)) \right]. \tag{6}$$

To control the drift term, we consider a region $\mathcal{N}_\Omega := \{\theta : \|\theta - \theta_0\| \le \Omega\}$ in which the score function is locally stable, i.e.,

$$\|\varphi_\theta(\tau) - \varphi_{\theta_0}(\tau)\| \le \kappa\|\theta - \theta_0\| \qquad \text{for all } \theta \in \mathcal{N}_\Omega,\ \tau \in \mathcal{T}, \tag{7}$$

with $\kappa\Omega \ll 1$, and $L = \sup_{\tau\in\mathcal{T}} \|\varphi_{\theta_0}(\tau)\|$ bounded. This is the standard lazy regime in NTK theory (Jacot et al., 2018; Chizat et al., 2019), and recent empirical work (Malladi et al., 2023) indicates that LLM fine-tuning often resides within this regime, justifying the use of a locally stable φ_{θ_0} as a first-order approximation.

Since $|R(\tau) - \bar{p}_\theta| \le 1$, Eq. (7) bounds the second term by $\frac{\kappa}{\sigma_\theta}\|\delta\|$. The first term becomes:

$$\begin{aligned}\mathbb{E}_{\pi_\theta}\left[\frac{R(\tau)-\bar{p}_\theta}{\sigma_\theta}\varphi_{\theta_0}\right] &= \frac{1}{\sigma_\theta}\left(\sum_{\tau\in S}\pi_\theta(\tau)\varphi_{\theta_0}(\tau) - \bar{p}_\theta\sum_{\tau\in\mathcal{T}}\pi_\theta(\tau)\varphi_{\theta_0}(\tau)\right)\\ &= \frac{1}{\sigma_\theta}\left(\sum_{\tau\in S}\pi_\theta(\tau)\varphi_{\theta_0}(\tau) - \bar{p}_\theta\sum_{\tau\in\mathcal{T}}\pi_\theta(\tau)(\varphi_{\theta_0}(\tau) - \varphi_\theta(\tau))\right),\end{aligned} \tag{8}$$

where we used $\sum_{\tau\in\mathcal{T}}\pi_\theta(\tau)\varphi_\theta(\tau) = 0$. The sum over $\mathcal{T}$ is bounded by $\kappa\|\delta\|$, leading to $g_{\text{GRPO}}(\theta) = \frac{1}{\sigma_\theta}\sum_{\tau\in S}\pi_\theta(\tau)\varphi_{\theta_0}(\tau) + r_1$, with $\|r_1\| \le \frac{2\kappa}{\sigma_\theta}\|\delta\|$.

To evaluate the sum over S, we expand the density ratio. For any response τ, we consider the straight-line path $\theta_s := \theta_0 + s\delta$ for $s \in [0,1]$. Then, the log-density ratio satisfies:

$$\log\pi_\theta(\tau) - \log\pi_{\theta_0}(\tau) = \int_0^1 \langle\delta, \varphi_{\theta_s}(\tau)\rangle ds. \tag{9}$$

Applying Eq. (7), the integrand satisfies $\langle\delta,\varphi_{\theta_s}(\tau)\rangle = \langle\delta,\varphi_{\theta_0}(\tau)\rangle + \epsilon_s$ where $|\epsilon_s| \le \kappa s\|\delta\|^2$. Integrating over $s \in [0,1]$, we obtain $\log(\pi_\theta/\pi_{\theta_0}) = \langle\delta,\varphi_{\theta_0}(\tau)\rangle + \epsilon_\tau$ with $|\epsilon_\tau| \le \frac{\kappa}{2}\|\delta\|^2$. Consequently, $\pi_\theta(\tau) = \pi_{\theta_0}(\tau)(1 + \langle\delta,\varphi_{\theta_0}(\tau)\rangle + \rho_\tau)$ where $|\rho_\tau| \le C\|\delta\|^2$ follows from the Taylor expansion of the exponential function. C is determined by the maximum curvature of the log-likelihood in $\mathcal{N}_\Omega$ (detailed in Appendix D.1). Substituting this into the sum over S:

$$\begin{aligned}\sum_{\tau\in S}\pi_\theta(\tau)\varphi_{\theta_0}(\tau) &= \sum_{\tau\in S}\pi_{\theta_0}(\tau)\varphi_{\theta_0}(\tau) + \left(\sum_{\tau\in S}\pi_{\theta_0}(\tau)\varphi_{\theta_0}(\tau)\varphi_{\theta_0}(\tau)^\top\right)\delta + \sum_{\tau\in S}\pi_{\theta_0}(\tau)\rho_\tau\varphi_{\theta_0}(\tau)\\ &= \bar{p}_0 v_R + \bar{p}_0 M_S\delta + r_2,\end{aligned} \tag{10}$$

where $M_S = \mathbb{E}_{\pi_{\theta_0}}[\varphi_{\theta_0}\varphi_{\theta_0}^\top \mid S]$ and $\|r_2\| \le CL\bar{p}_0\|\delta\|^2$. Similarly, expanding $\bar{p}_\theta$ yields:

$$\bar{p}_\theta = \sum_{\tau\in S}\pi_\theta(\tau) = \bar{p}_0 + \bar{p}_0 v_R^\top\delta + r_3, \quad \text{where } |r_3| \le C\bar{p}_0\|\delta\|^2. \tag{11}$$

Combining these to analyze the deviation from the direction $\frac{\bar{p}_\theta}{\sigma_\theta}v_R$:

$$\begin{aligned}g_{\text{GRPO}}(\theta) - \frac{\bar{p}_\theta}{\sigma_\theta}v_R &= \frac{1}{\sigma_\theta}(\bar{p}_0 v_R + \bar{p}_0 M_S\delta + r_2) - \frac{1}{\sigma_\theta}(\bar{p}_0 + \bar{p}_0 v_R^\top\delta + r_3)v_R + r_1\\ &= \frac{\bar{p}_0}{\sigma_\theta}(M_S - v_R v_R^\top)\delta + \frac{r_2}{\sigma_\theta} - \frac{r_3 v_R}{\sigma_\theta} + r_1.\end{aligned} \tag{12}$$

Defining $K_S = M_S - v_R v_R^\top$, which is the covariance of the score over successful responses at initialization, the norm of the deviation is:

$$\left\|g_{\text{GRPO}}(\theta) - \frac{\bar{p}_\theta}{\sigma_\theta}v_R\right\| \le \underbrace{\frac{\bar{p}_0\|K_S\|_{\text{op}} + 2\kappa}{\sigma_\theta}\|\delta\|}_{\text{first-order change}} + \underbrace{\frac{C(L + \|v_R\|)\bar{p}_0}{\sigma_\theta}\|\delta\|^2}_{\text{second-order change}}. \tag{13}$$

Since $\bar{p}_0 \le 1$ and $\|v_R\| \le L$, the second-order term is at most $\frac{2CL}{\sigma_\theta}\|\delta\|^2$.

Two features of Eq. (13) are worth noting. First, the scale of the leading direction, $\frac{\bar{p}_\theta}{\sigma_\theta} = \sqrt{\frac{\bar{p}_\theta}{1-\bar{p}_\theta}}$, stays moderate across the intermediate regime of success rates. For $\bar{p}_\theta \in [0.2, 0.8]$ it ranges only

from about 0.5 to 2.0, avoiding the vanishing or exploding gradients typical of extreme reward scenarios. Second, both error terms carry a factor $1/\sigma_\theta$, so a larger reward variance yields a smaller absolute deviation from the leading direction, while a small σ_θ makes the update scale sensitive to slight perturbations of $\bar{p}_\theta$. We therefore read Eq. (13) as a scale-based account of why the naturally occurring high-variance regime of RLVR can support a stable, near-constant drift along v_R, and we regard high reward variance as a plausible explanation for the observed linearity. This reading is consistent with the behavior near entropy collapse reported in Appendix G, where the reward spread shrinks and the trajectory correspondingly loses linearity.

Off-policy sampling. In practical GRPO, responses are sampled from an old policy $\pi_{\theta_{\text{old}}}$ and reused for several actor updates, with each term reweighted by the importance ratio $\pi_\theta(\tau)/\pi_{\theta_{\text{old}}}(\tau)$. This is an importance-sampling transformation for off-policy reuse, so in expectation

$$\mathbb{E}_{\tau\sim\pi_{\theta_{\text{old}}}}\left[\frac{\pi_\theta(\tau)}{\pi_{\theta_{\text{old}}}(\tau)}\left(\frac{R(\tau)-\bar{p}_\theta}{\sigma_\theta}\right)\varphi_\theta(\tau)\right] = g_{\text{GRPO}}(\theta), \tag{14}$$

which recovers Eq. (4). Since our analysis is formulated at the level of the expected update direction rather than a finite-sample estimator, the old-policy ratio does not change the object being analyzed or the resulting bound.

Clipping and length normalization. Two further implementation details modify individual terms of the sum but not its structure. Denote the token-level importance ratio $w_{i,t}(\theta) = \pi_\theta(y_{i,t} \mid x, y_{i,<t})/\pi_{\theta_{\text{old}}}(y_{i,t} \mid x, y_{i,<t})$ and let $m_{i,t}(\theta) \in \{0,1\}$ be the clipping mask: $m_{i,t} = 0$ (term removed) when $\hat{A}_i \geq 0$ and $w_{i,t} > 1+\epsilon$, or when $\hat{A}_i < 0$ and $w_{i,t} < 1-\epsilon$; otherwise $m_{i,t} = 1$. The difference between the clipped and unclipped update is

$$g_{\text{unclip}}(\theta) - g_{\text{clip}}(\theta) = \mathbb{E}_{\pi_{\theta_{\text{old}}}}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{L_i}\sum_t \hat{A}_i\,(1-m_{i,t}(\theta))\,w_{i,t}(\theta)\,\varphi_{\theta,i,t}\right]. \tag{15}$$

At $\theta = \theta_{\text{old}}$ every ratio equals one, the clipping condition is never triggered, and $m_{i,t} = 1$ throughout, so $g_{\text{clip}} = g_{\text{unclip}}$. Away from θ_{old}, each removed term satisfies $\|\hat{A}_i w_{i,t}\varphi_{\theta,i,t}/L_i\| \leq |\hat{A}_i| w_{i,t} L/L_i$, so Eq. (15) is bounded by the corresponding length-weighted expectation $\mathbb{E}[|\hat{A}_i|\, w_{i,t}\, L/L_i\, \mathbf{1}\{m_{i,t} = 0\}]$, which vanishes when θ is near θ_{old}.

Length normalization changes the quantities in the preceding unnormalized calculation as follows. Writing $\ell_\tau := |\tau|$, let

$$\begin{aligned} p_\theta^{(\ell)} &:= \mathbb{E}_{\pi_\theta}\left[\frac{R(\tau)}{\ell_\tau}\right], \qquad p_0^{(\ell)} := \mathbb{E}_{\pi_{\theta_0}}\left[\frac{R(\tau)}{\ell_\tau}\right], \\ \sigma_\theta^{(\ell)} &:= \sqrt{p_\theta^{(\ell)}(1-p_\theta^{(\ell)})}, \qquad v_R^{(\ell)} := \frac{1}{p_0^{(\ell)}}\mathbb{E}_{\pi_{\theta_0}}\left[\frac{R(\tau)}{\ell_\tau}\varphi_{\theta_0}(\tau)\right]. \end{aligned} \tag{16}$$

The full length-normalized counterpart of Eq. (8) is

$$\begin{aligned} \mathbb{E}_{\pi_\theta}\left[\frac{R(\tau)-p_\theta^{(\ell)}}{\sigma_\theta^{(\ell)}}\frac{\varphi_{\theta_0}(\tau)}{\ell_\tau}\right] &= \frac{1}{\sigma_\theta^{(\ell)}}\left(\sum_{\tau\in S}\pi_\theta(\tau)\frac{\varphi_{\theta_0}(\tau)}{\ell_\tau} - p_\theta^{(\ell)}\sum_{\tau\in\mathcal{T}}\pi_\theta(\tau)\frac{\varphi_{\theta_0}(\tau)}{\ell_\tau}\right) \\ &= \frac{1}{\sigma_\theta^{(\ell)}}\left(\sum_{\tau\in S}\pi_\theta(\tau)\frac{\varphi_{\theta_0}(\tau)}{\ell_\tau} - p_\theta^{(\ell)}\sum_{\tau\in\mathcal{T}}\pi_\theta(\tau)\big(\varphi_{\theta_0}(\tau)-\varphi_\theta(\tau)\big)\right. \\ &\qquad \left. + p_\theta^{(\ell)}\sum_{\tau\in\mathcal{T}}\pi_\theta(\tau)\left(1-\frac{1}{\ell_\tau}\right)\varphi_{\theta_0}(\tau)\right), \end{aligned} \tag{17}$$

The first sum, over successful responses, admits the same density-ratio expansion as above. Its leading direction is therefore still controlled by the same type of covariance and density-ratio remainder terms, but the fixed direction is now the length-weighted quantity $\frac{p_\theta^{(\ell)}}{\sigma_\theta^{(\ell)}} v_R^{(\ell)}$. The second sum, containing $\varphi_{\theta_0} - \varphi_\theta$, is bounded in exactly the same way as the score-drift term above. The zero-score identity

is applied only to the unnormalized score φ_θ, which leaves the third, length-weighted sum. This additional sum satisfies

$$\left\| \sum_{\tau \in \mathcal{T}} \pi_\theta(\tau) \left(1 - \frac{1}{\ell_\tau}\right) \varphi_{\theta_0}(\tau) \right\| \leq L \, \mathbb{E}_{\pi_\theta} \left[1 - \frac{1}{\ell_\tau}\right]. \tag{18}$$

Thus the third term contributes the bounded correction

$$\frac{p_\theta^{(\ell)} L}{\sigma_\theta^{(\ell)}} \mathbb{E}_{\pi_\theta} \left[1 - \frac{1}{\ell_\tau}\right], \tag{19}$$

and the successful-response expansions use $p_0^{(\ell)}$, $v_R^{(\ell)}$, and the corresponding length-weighted second-moment matrix. Hence length normalization changes the fixed direction and adds a bounded correction, but the overall conclusion of linearity is unchanged in substance.

Finite-group normalization. Eq. (4) normalizes by the population statistics $\bar{p}_\theta$ and σ_θ, whereas GRPO normalizes within a group of G responses sampled for the same prompt. This makes the normalization prompt-conditional: writing $p_x = \Pr(R = 1 \mid x)$ and $\sigma_x = \sqrt{p_x(1-p_x)}$, the group-relative advantage is normalized by σ_x instead of σ_θ. Accordingly, the quantities v_R, K_S, and $\bar{p}_0$ become prompt-conditional statistics v_x, K_x, and p_x, and Eq. (4) carries an additional expectation over prompts $x \sim \mathcal{D}$. Repeating the derivation under this refinement gives the leading term

$$\mathbb{E}_{x \sim \mathcal{D}} \left[\frac{p_x}{\sigma_x} v_x\right], \tag{20}$$

with the deviation bounded by the prompt-averaged counterpart of Eq. (13). Because the expectation over x is taken in the population update, Eq. (20) is again a fixed direction, so prompt-dependent normalization does not destroy the alignment with a fixed direction. It only refines the definition of that direction, replacing $\frac{\bar{p}_\theta}{\sigma_\theta} v_R$ by its prompt-conditional average.

D.1 Detailed derivation of C

In this section, we provide a detailed derivation of the second-order constant C used above. Our objective is to bound the residual ρ_τ in the density ratio expansion $\pi_\theta(\tau)/\pi_{\theta_0}(\tau) = 1 + \varphi_{\theta_0}(\tau)^\top \delta + \rho_\tau$.

Taylor expansion with Lagrange remainder. Consider the log-density difference $x_\tau := \log \pi_\theta(\tau) - \log \pi_{\theta_0}(\tau)$. By the second-order Taylor theorem, there exists a point ξ_τ strictly between 0 and x_τ such that:

$$e^{x_\tau} = 1 + x_\tau + \frac{1}{2} x_\tau^2 e^{\xi_\tau}. \tag{21}$$

The term e^{ξ_τ} captures the local curvature of the exponential mapping. In the lazy regime, we operate within the neighborhood $\mathcal{N}_\Omega$, where the displacement $\|\delta\| \leq \Omega$ is small. Since $|x_\tau| \leq L\|\delta\| + \frac{\kappa}{2}\|\delta\|^2$, it follows that $x_\tau \to 0$ as $\Omega \to 0$, implying e^{ξ_τ} is well-behaved and close to unity. We define a uniform upper bound for this curvature across $\mathcal{N}_\Omega$ as:

$$e^\eta := \sup_{\theta \in \mathcal{N}_\Omega, \tau \in \mathcal{T}} e^{\xi_\tau} \leq \exp\left(L\Omega + \frac{\kappa}{2}\Omega^2\right). \tag{22}$$

Bounding the pointwise residual ρ_τ. Combining the expansion with our control on x_τ in Eq. (9), where $x_\tau = \varphi_{\theta_0}(\tau)^\top \delta + \epsilon_\tau$ with $|\epsilon_\tau| \leq \frac{\kappa}{2}\|\delta\|^2$, we have:

$$\rho_\tau = \epsilon_\tau + \frac{1}{2}\left(\varphi_{\theta_0}(\tau)^\top \delta + \epsilon_\tau\right)^2 e^{\xi_\tau}. \tag{23}$$

Taking absolute values and applying the triangle inequality:

$$\begin{aligned} |\rho_\tau| &\leq |\epsilon_\tau| + \frac{1}{2}\left(|\varphi_{\theta_0}(\tau)^\top \delta| + |\epsilon_\tau|\right)^2 e^{\xi_\tau} \\ &\leq \frac{\kappa}{2}\|\delta\|^2 + \frac{1}{2}\left(L\|\delta\| + \frac{\kappa}{2}\|\delta\|^2\right)^2 e^\eta. \end{aligned} \tag{24}$$

Neglecting the higher-order terms $\mathcal{O}(\|\delta\|^3)$, the residual is bounded by:

$$|\rho_\tau| \leq \frac{1}{2}(L^2 e^\eta + \kappa)\|\delta\|^2. \tag{25}$$

Explicit definition of C. Comparing with the bound $|\rho_\tau| \le C\|\delta\|^2$ used above, the constant C is explicitly given by:

$$C := \frac{1}{2}\left(L^2 e^\eta + \kappa\right), \tag{26}$$

where $e^\eta \le \exp\left(L\Omega + \frac{\kappa}{2}\Omega^2\right)$ uniformly bounds the exponential curvature in $\mathcal{N}_\Omega$. In the lazy regime where $\kappa\Omega \ll 1$ and $L\Omega$ is moderate, $e^\eta = 1 + \mathcal{O}(L\Omega)$, so C is a finite constant depending only on the three external quantities L, Ω, and κ, and is independent of the model dimension d. In particular, when $\Omega \to 0$ we recover $C \to \frac{1}{2}(L^2 + \kappa)$.

E From weight linearity to activation and output linearity

Linear weight trajectories do not necessarily produce linear outputs in a nonlinear Transformer. However, the residual structure provides an intuitive route for approximately linear changes to propagate: each sublayer passes along its input and adds a correction whose change during RL can be dominated by its first-order response to input and weight changes. Here we consider fixed token sequences, so that changes arise from training rather than from different input contexts.

Write a sublayer as $x_{\ell+1,t} = x_{\ell,t} + F_\ell(x_{\ell,t}, W_{\ell,t})$, where F_ℓ represents an attention or MLP layer, and $W_{\ell,t}$ collects its trainable parameters. Define changes from the initial checkpoint as $\delta x_{\ell,t} = x_{\ell,t} - x_{\ell,0}$ and $\delta W_{\ell,t} = W_{\ell,t} - W_{\ell,0}$. The change in the layer output can be represented as

$$\delta y_{\ell,t} = (I + J_{x,\ell})\delta x_{\ell,t} + J_{W,\ell}\delta W_{\ell,t} + R_{\ell,t}, \tag{27}$$

where $J_{x,\ell}$ and $J_{W,\ell}$ are the Jacobians with respect to the inputs and weights, and $R_{\ell,t}$ collects higher-order terms. Thus, each sublayer transmits the incoming change from previous layers and adds the effect of its own parameter changes. When cumulative perturbations remain relatively small and local curvature is bounded, approximately linear changes can accumulate through these first-order responses.

We investigate the down-projection layer of the final MLP as a particularly stringent case. Although its input change δx_t has accumulated the changes from all preceding layers, it accounts for only 7.7% of the magnitude of the original x_t on average throughout RL training. Because the output is $y_t = W_t x_t$, its change decomposes exactly as

$$\delta y_t = W_0 \delta x_t + \delta W_t x_0 + \delta W_t \delta x_t. \tag{28}$$

The first two terms reflect the first-order effects of cumulative input changes and weight updates, while the last term represents their second-order interaction. As shown in Figure 4(b), because both δx_t and δW_t are relatively small, the second-order term $\delta W_t \delta x_t$ remains centered near zero, yielding nearly linear output dynamics.

The same first-order reasoning extends to nonlinear components, although their expansions differ. In attention, changes to $QK^\top$ contain first-order query and key contributions as well as the second-order interaction $\delta Q \delta K^\top$. For small perturbations, softmax produces a first-order redistribution of attention probabilities, while multiplication by the values introduces another interaction between two changes. LayerNorm and SiLU likewise admit first-order expansions when perturbations are sufficiently small. Consistent with this account, Figure 4(c) shows that activation trajectories become increasingly linear in deeper layers and that the output logits attain particularly high linearity.

F Additional Results on Extrapolation

F.1 Empirical guidelines for weight-space extrapolation hyperparameters

Based on our extensive experimental findings, the performance of weight-space extrapolation relies on the appropriate selection of the anchor checkpoints (t_0 and t_1) and the extrapolation ratio (β). Assuming a total available training budget of N steps, we summarize our empirical guidelines for these hyperparameters as follows:

- **Initial Anchor t_0:** The model performance is generally not highly sensitive to the exact choice of the starting step t_0. In practice, we recommend setting t_0 to approximately 20% to 30% of the total budget N. This allows the model to bypass the initial unstable phase of training and provides a reliable starting point for estimating the optimization trajectory.

Table 7: **Periodic Re-grounding Hyperparameter Configurations.** Values are reported for each training budget s corresponding to Table 2.

Training Budget (s)	m	n
200	100	100
400	300	600
800	100	100
1200	100	100

- **Second Anchor t_1:** To capture the most mature weight update direction and maximize the utilization of the available training dynamics, t_1 should be set as late as possible. Therefore, we recommend setting $t_1 = N$, utilizing the final checkpoint of the training budget.
- **Extrapolation Ratio β:** The optimal scaling factor β (which directly determines the target step t) is highly task-specific. As discussed in Section 5.2 and illustrated in Figure 5, while a moderate β strictly improves performance, an excessively large β can amplify trajectory estimation errors. Consequently, β should be carefully tuned on a validation set according to the specific downstream task and dataset characteristics.

F.2 Periodic re-grounding

Periodic Re-grounding mechanism serves as an implementation strategy for refreshing the local direction of the trajectory once extrapolation becomes unreliable. Concretely, each cycle uses m genuine RL steps to estimate a local direction and then projects this direction for n additional virtual steps. This design preserves the efficiency benefits of extrapolation while reducing the error accumulation associated with long-horizon projection.

We select (m, n) using AIME24 only, considering schedules with $n/m \in [1, 5]$. After selection, the resulting configuration is fixed and evaluated on the other benchmarks without further tuning. The specific configurations selected for each training budget in Section 5.3 are listed in Table 7.

Difference between Periodic Re-grounding and Fine-tuning LR: We investigate the fundamental differences between our proposed Periodic Re-grounding mechanism and the naive approach of scaling the learning rate (LR). Naive LR scaling inevitably fails due to severe gradient noise; specifically, the gradients exhibit a critically low correlation (< 0.03) with the dominant update direction and suffer from high variance. Consequently, increasing the learning rate directly amplifies this noise, leading to extreme training instability and frequent crashes. Empirically, scaling the LR from 1e-6 to 1e-5 results in a severe performance degradation, with AIME24 accuracy drastically dropping from 0.42 to 0.32. To overcome these limitations, Periodic Re-grounding mechanism achieves stable acceleration through an identify-then-extrapolate strategy. Rather than amplifying noisy updates, it first calibrates the model using 20 actual reinforcement learning (RL) steps to reliably establish a stable linear direction. Following this calibration, it executes a long-step extrapolation by projecting 100 steps along this identified trajectory. This approach effectively circumvents gradient noise, yielding a >6x training speedup while strictly preserving model accuracy.

G Implications for Understanding RLVR

Elicitation vs. Emergence in RLVR. A central debate in reasoning-oriented LLMs concerns whether RLVR instills novel cognitive capabilities or merely elicits pre-existing ones. Proponents of capability emergence argue that verifiable rewards facilitate the composition of atomic reasoning primitives to solve unseen tasks (Wen et al., 2026; Wang et al., 2026). Conversely, the "Invisible Leash" hypothesis (Wu et al., 2026; Yue et al., 2025) contends that RLVR is strictly bounded by the base model's pre-trained manifold. This elicitation-only view is empirically supported by test-time scaling studies, where majority voting often matches RLVR peak performance (Wang et al., 2023; Karan & Du, 2026). Our mechanistic findings on RLVR linearity lend rigorous mathematical support to the elicitation hypothesis: rather than executing complex nonlinear searches for novel behaviors,

current RLVR algorithms predominantly amplify and up-weight dominant reasoning trajectories established in the earliest stages of training.

This perspective also suggests a concrete direction for future RLVR algorithms. RL training must promote reasoning patterns that are initially rare under the base model. New skills are more likely to arise when low-probability but valuable traces are surfaced and selectively reinforced. This could be achieved by expanding exploration during rollout, e.g., with higher temperatures and larger rollout budgets, together with reweighting schemes that favor novel successful samples, or by replaying rare positive traces that standard on-policy updates might otherwise wash out. More broadly, escaping the current linear, elicitation-dominated regime may require RLVR algorithms that explicitly combine exploration, selective amplification, and memory.

Training Stability & Acceleration. Weight linearity also suggests a promising route toward more efficient RLVR optimization. If training follows a linear trajectory in parameter space, then small updates obtained from different batches and successive steps may be safely accumulated into a much larger effective gradient update. This opens up the possibility of replacing many fine-grained parameter updates with less frequent but larger, more compute-efficient ones. From this perspective, a key research question is to characterize the scaling law among learning rate, batch size, and rollout budget: understanding how these quantities co-vary within the linear regime may reveal operating points that simultaneously improve stability and accelerate training. More broadly, the linear structure we observe suggests that RLVR may be amenable to principled update accumulation strategies that better exploit available compute without sacrificing optimization reliability.

Instability in the Later Stages. As shown in Table 1, output-space extrapolation outperforms standard RL across three benchmarks. A plausible explanation is that output-space extrapolation preserves the stable improvement direction established in earlier training, while avoiding some of the degradation that can arise in late-stage RLVR, such as entropy collapse or over-fitting. In this sense, output-space extrapolation acts as a lightweight way to continue the policy's functional trajectory without incurring the full cost—or instability—of additional RL updates.

Our results also suggest a new perspective on RLVR instability. In our experiments, weight evolution loses linearity near entropy collapse. This raises two possibilities: either continued logit amplification drives entropy collapse and then breaks the linear regime, or linearity breaks first and serves as an early warning signal for instability. In the latter case, extrapolation could be not only predictive but also stabilizing, by keeping training on the pre-collapse trajectory and away from unstable regions. This points to a broader direction: understanding the interplay between logit growth, entropy dynamics, and trajectory linearity may help anticipate and mitigate RLVR instability.

H LIMITATIONS

While our study provides robust empirical evidence and analysis for the linear regime in RLVR, it presents several limitations that point to promising directions for future research:

- **Scope of Reward Modalities:** Our analysis strictly focuses on Reinforcement Learning with Verifiable Rewards (RLVR) in domains with objective correctness, such as mathematics and coding. It remains an open question whether similar striking linear dynamics dominate in standard RLHF pipelines that rely on learned, noisy reward models, or in open-ended text generation tasks where the reward landscape is fundamentally different.
- **Extrapolation Horizon and Hyperparameter Tuning:** Although weight-space extrapolation yields significant compute savings, the optimal extrapolation ratio (β) remains highly task-specific. As demonstrated, pushing the projection horizon too far without periodic re-grounding amplifies trajectory estimation errors. Consequently, identifying the optimal hyperparameter configuration currently requires empirical tuning on a validation set, adding slight overhead to the otherwise compute-saving method.
- **Model Scale and Algorithms:** We evaluated models up to 32B parameters and focused primarily on memory-efficient, critic-free RL algorithms (e.g., GRPO, REINFORCE++, and GSPO). While the linear regime persists consistently across these configurations, verifying these dynamics on frontier-scale models (e.g., 100B+ parameters) or under standard actor-critic PPO frameworks is deferred to future work due to computational constraints.

- **Curriculum Learning:** The gradient calculation uses a fixed prompt distribution $\mathcal{D}$. Under Curriculum Learning, the dynamic evolution of the data distribution $\mathcal{D}_t$ may shift the dominant optimization subspace. We hypothesize that such trajectories exhibit *piecewise linearity* within stable curriculum stages rather than strict global linearity. Characterizing this geometry under non-stationary distributions remains an important direction for future work.