Not All Steps are Informative: On the Linearity of LLMs' RLVR Training

Anonymous ACL submission

Abstract

Reinforcement learning with verifiable rewards (RLVR) has become a central component of large language model (LLM) post-training. Unlike supervised fine-tuning (SFT), RLVR lets an LLM generate multiple candidate solutions and reinforces those that lead to a verifiably correct final answer. However, in practice, RLVR often requires thousands of training steps to reach strong performance, incurring substantial computation largely attributed to prolonged exploration. In this work, we make a surprising observation: during RLVR, LLMs evolve in a strongly linear manner. Specifically, both model weights and model output log-probabilities exhibit strong linear correlations with RL training steps. This suggests that RLVR predominantly amplifies trends that emerge early in training, rather than continuously discovering new behaviors throughout the entire optimization trajectory. Motivated by this linearity, we investigate whether future model states can be predicted from intermediate checkpoints via extrapolation, avoiding continued expensive training. We show that Weight Extrapolation produces models with performance comparable to standard RL training while requiring significantly less computation. Moreover, Logits Extrapolation consistently outperforms continued RL training on all four benchmarks by extrapolating beyond the step range where RL training remains stable.

1 Introduction

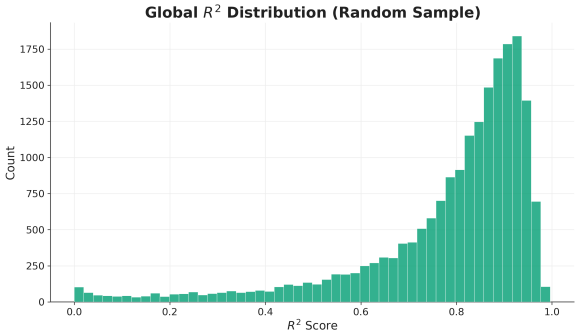
Large Language Models (LLMs) have demonstrated remarkable capabilities in complex reasoning tasks, largely driven by the adoption of Reinforcement Learning with Verifiable Rewards (RLVR) (Jaech et al., 2024; Lambert et al., 2024; DeepSeek-AI et al., 2025; Yang et al., 2025). By leveraging outcome-based supervision—such as the correctness of a mathematical solution or the execution of code (Shao et al., 2024; Le et al., 2022; Wang et al., 2023; Hu et al., 2025b; Face, 2024). RLVR has proven to be a highly effective approach for boosting reasoning performance while minimizing the forgetting of old knowledge (Chen et al., 2025; Shenfeld et al., 2025).

Despite its efficacy, the current RLVR paradigm remains highly resource-intensive, severely limiting scalability. This inefficiency mainly stems from two factors. First, RLVR typically requires a large number of training steps to reach strong performance. For example, training R1-Zero from a base model commonly needs on the order of 8,000 steps (DeepSeek-AI et al., 2025) to achieve the desired capability. Second, the rollout trajectories tend to become progressively longer as training proceeds (i.e., the model learns to generate longer reasoning chains of thoughts) (Zhang et al., 2025; Li et al., 2025). As a result, the wall-clock time per step can increase dramatically, from only a few minutes early in training to tens of minutes later, further amplifying the overall compute cost. A concrete example illustrates the magnitude of this overhead: even for a relatively small 1.5B model, training DeepSeek-R1-Distill-Qwen-1.5B on DeepScaleR requires approximately 3,800 A100 GPU-hours (about 5 days on 32 A100s) (Luo et al.).

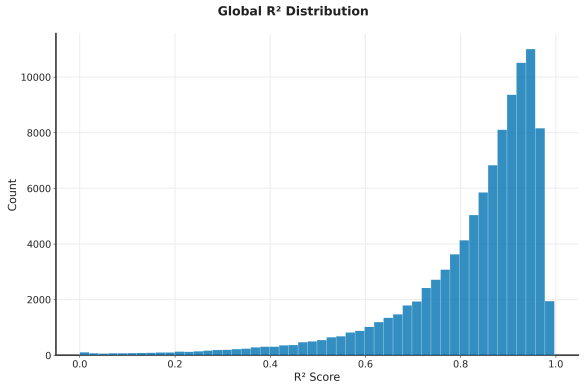
In this work, we argue that most of the training steps in the current RLVR algorithms are not informative, which is part of the reason for the computational inefficiency of RLVR. Our key insight is based on a surprising observation: during RLVR, the per-step change in model weights and model outputs (for example, token log-probabilities for a given input sequence) evolve approximately linearly over RL training steps.

> ***"During RLVR training, LLM weights and outputs exhibit strong linear correlations with training step."***

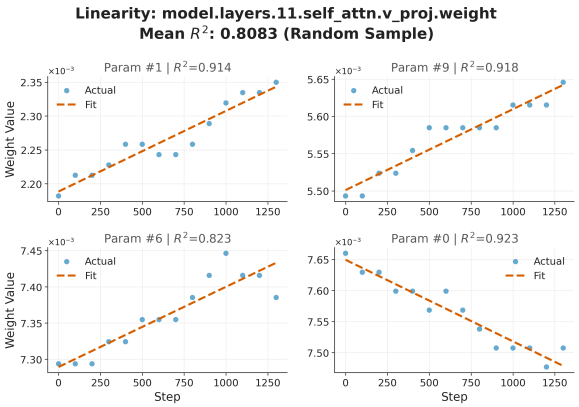
To quantify and validate this phenomenon, we conducted RLVR training on math problems, and then performed linearity analysis on both model



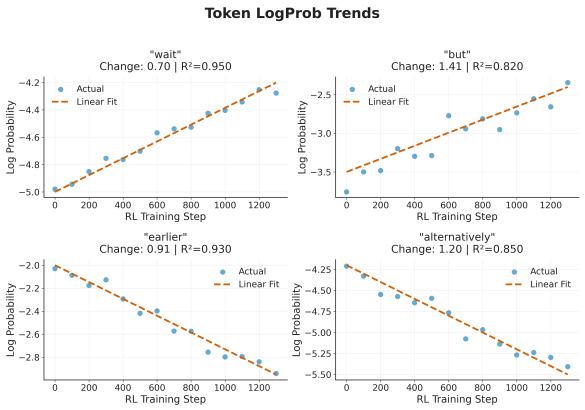
(a) R^2 distribution of weights.



(b) R^2 distribution of token log-probabilities.



(c) Examples of weight dynamics.



(d) Examples of token log-probability dynamics.

Figure 1: Linearity analysis for model weights and outputs during RLVR training. (a) and (b) show the distributions of R^2 for weight and token log-probabilities, respectively. Both distributions are concentrated around 0.9, indicating strong linearity. (c) plots the trajectories of four randomly selected weights, and (d) shows token log-probability changes at four example positions. The log-probabilities of "wait" and "but" increase over RL steps, suggesting more reflection and revision, whereas those of "earlier" and "alternatively" decrease, indicating reduced need for backtracking and branching.

weights and outputs.

For weight linearity, we performed linear regression of weights at all checkpoints against training steps. We observe clear linear weight dynamics: as training progresses, the majority of weights change in a highly linear fashion. For example, as shown in Fig. 1c, weight #1, #6, and #9 exhibit near-linear increases, whereas parameter #0 decreases nearly linearly; Fig. 1c shows the R^2 distribution of all weights, excluding those that are unchanged during training. We can see that more than 80% of the weights achieve $R^2 > 0.7$, indicating strong weight linearity.

We also performed controlled checkpoint-probing to analyze the linearity of model outputs. Specifically, we selected a set of queries and their solution trajectories as probes. For each checkpoint, we computed the log-probabilities of each token, conditional on all previous tokens. As shown in Fig. 1a and 1d, there is also a strong linear correlation between model predicted log-probabilities and training steps. We also observed linearity of other model outputs, such as logits, and intermediate activations, whose results are left in Appendix A.2.

We validate the generality of this phenomenon across a wide range of settings, we repeat our experiments, over different base models (DeepSeek-R1-Distill-series (DeepSeek-AI et al., 2025) and Open-Nemotron-1.5B (Moshkov et al., 2025)), RL training paradigms (GRPO (Shao et al., 2024), GSPO (Zheng et al., 2025) and Reinforce++ (Hu et al., 2025a)), we consistently observe statistically significant step-wise linear trends in both token model weights and outputs. Collectively, these linearities imply that the marginal information gain of additional RLVR steps is even lower than previously thought: the model is largely continuing a predictable trajectory rather than acquiring new

behaviors. This, in turn, suggests an opportunity for efficiency—by exploiting the linear structure in both logprob and weight dynamics, we can reduce the training computation required to reach the same level of performance.

To leverage the approximately linear dynamics observed during training, we propose three acceleration schemes, including direct extrapolations on weights and logits (Logit Extrapolation and Weight Extrapolation), and an iterative approach that alternates weight extrapolation with actual RL traininig (RL-Extra). Our experimental results show that Weight Extrapolation can extrapolate for up to **600** without performance degradation compared with actually training the model for the same RL steps. Also, by Logit Extrapolation we extrapolate beyond the step that the model can be stably trained, and observe up to a **3%** improvement over standard RL training across multiple experiments.

However, when the extrapolation horizon exceeds 1,000 training steps, performance begins to degrade, suggesting that the linearity assumption breaks down over long ranges. To address this, we introduce RL-Extra, which alternates between short bursts of RL training and weight extrapolation. The short RL phase periodically recalibrates the gradients and corrects extrapolation errors. Across a range of settings, RL-Extra matches the performance of standard RL training while delivering up to a **6.1**$\times$ speedup.

In summary, our contributions are as follows:

- We identify and theoretically explain the strong linearity in weight updates and model output token log-probabilities across training steps, validating its universality across diverse models and algorithms.
- We propose Weight Extrapolation and Logit Extrapolation to estimate future model states without expensive rollouts, reducing computational costs by 800 RL steps and achieving up to a **3%** performance improvement over standard baselines, respectively.
- We introduce RL-Extra, an accelerated training paradigm that interleaves gradient updates with weight extrapolation, delivering up to a **6.1**$\times$ wall-clock speedup.

2 Background and Related Works

2.1 Preliminaries in RLVR

RLVR has emerged as a critical part of LLM post-training. By leveraging deterministic, rule-based binary feedback, RLVR optimizes LLM performance without the noisy proxies inherent in Reinforcement Learning from Human Feedback (RLHF) (Bai et al., 2022; Ouyang et al., 2022). This approach enhances transparency and efficiency, proving particularly potent in domains demanding objective correctness (Uesato et al., 2022; Shao et al., 2024; Le et al., 2022), especially mathematical reasoning.

Recent advancements have demonstrated the efficacy of this paradigm. Guo et al. (2025) introduced *DeepSeek-R1*, which utilizes RLVR to significantly incentivize reasoning capabilities in LLMs without extensive supervised fine-tuning. A core component of modern RLVR is Group Relative Policy Optimization (GRPO), proposed in *DeepSeekMath* (Shao et al., 2024). GRPO eliminates the need for a value network by generating multiple responses per prompt, scoring them with a deterministic function, and then using the group-normalized reward to update the LLM. Several variants of GRPO have been introduced to enhance its stability. For example, REINFORCE++ (Hu et al., 2025a) proposes Global Advantage Normalization to replace local group normalization in GRPO, eliminating the critic to reduce computation and correcting the bias introduced by per-prompt normalization in existing critic-free approaches. For improving the training stability of MOE models, GSPO (Zheng et al., 2025) is proposed, elevating the optimization granularity from token level to sequence level.

2.2 Mechanisms of RLVR

A series of recent research delves into the internal mechanisms of RLVR, analyzing how it enhances reasoning from the perspectives of capability boundaries and parameter dynamics.

Capability Boundaries and Effectiveness. A pivotal debate in RLVR is whether it instills new capabilities or merely elicits latent ones. (Mroueh, 2025) argues that RLVR with verifiable rewards implicitly incentivizes correct reasoning chains in base LLMs, even when only final answers are rewarded. However, (Wu et al., 2025) and (Yue et al., 2025) suggest an "Invisible Leash," indicating that RLVR may not escape the inherent capacity constraints of the pre-trained base model. Supporting this view, (Wang et al., 2022; Karan and Du, 2025) demonstrates that simple scaling of inference (e.g., majority voting or power sampling) can match RLVR performance, implying that RLVR

essentially optimizes the sampling distribution to align with the model's existing best-performance subspace rather than learning new knowledge from scratch. (Zhao et al., 2025) using only the top 20% highest-entropy tokens for RL updates yields better performance than updating on all tokens; moreover, discarding the remaining 80% low-entropy tokens can lead to further gains.

Our discoveries on RLVR linearity further points out the possibility that current RLVR algorithms only adjust the probabilies of frequent patterns that can be seen at the beginning of training.

Structure in Training Dynamics: Sparsity and Subspace. From the more detailed perspective of weight updates, previous works have reveals that RLVR updates exhibit distinct structural properties. Despite being trained with AdamW on full parameters, the weight updates in RLVR are often highly sparse. Mukherjee et al. (2025) observe that RL finetuning primarily updates small subnetworks (approximately **20%** of parameters) while leaving the majority of the model unchanged. From a geometric perspective, Zhu et al. (2025) proves that RLVR learns primarily along *non-principal directions* of the Hessian or feature space. This "Path Not Taken" suggests that RLVR refines the model by perturbing it in directions orthogonal to its principal components, thereby enhancing specific reasoning tasks without catastrophic forgetting of general knowledge. This sparsity in updates explains why RLVR can achieve significant performance gains with relatively low data requirements (Wang et al., 2025) and minimal interference with the model's core linguistic capabilities.

Another closely related area is Parameter-Efficient Fine-Tuning (PEFT). Hu et al. (2021) first demonstrated that low-rank matrix factorization can enable efficient adaptation of large language models, substantially reducing the number of trainable parameters while achieving performance comparable to full fine-tuning. Subsequent methods such as DoRA(Liu et al., 2024a), MiSS(Zhang et al., 2023b), and AdaLoRA (Zhang et al., 2023a) further refined the parameterization. PiSSA and MiLoRA (Meng et al., 2024; Liu et al., 2024b) introduced singular value decomposition (SVD)-based initialization. VeRA (Kopiczko et al., 2023) pushed compression more aggressively. By replacing the two per-layer low-rank matrices A and B in LoRA with *globally shared and frozen* random matrices, and training only two extremely lightweight scaling vectors b and d, the method uses diagonal matrices Λ_b and Λ_d to gate/scale rows and columns on a per-layer basis. This reduces the number of trainable parameters by an additional 10–30$\times$ without degrading performance, while introducing zero inference latency.

These works have demonstrated that the information gain in RLVR is limited by the number of sparse or low-rank weight updates. Our work verifies the limited information gain from another perspective. With the linearity of RLVR, only a small number of training steps are truly informative, restricting the amount of information instilled into the LLM during RLVR.

3 Linearity of RLVR Training

In this section, we examine the linearity of both model weights and output log-probability across diverse settings, encompassing various training data, base models, and rl algorithms. We further provide a theoretical explanation for these observations, which is counter-intuitive given the highly nonlinear nature of transformer-based architectures.

3.1 Linearity in Weights

To investigate the linearity of weight update during RLVR, we perform a linear regression analysis on the model parameters throughout the training process and calculate the coefficient of determination R^2. Specifically, we reproduce DeepScaleR (Luo et al.) training process by training a *Deepseek-R1-Distilled-Qwen-1.5B* base model on the *DeepScaleR-Preview* dataset (training details are provided in Appendix A.1). Given the vast parameter space in LLMs, we randomly sample (0.1%) of weights from each layer for analysis. We also exclude all weights that rarely change because of the precision of bfloat16 for computational stability and the easiness of analysis.

As illustrated in Fig 1a, the distribution of R^2 is heavily concentrated around 0.9, indicating a strong linearity in weight updates. We further analyze the average R^2 across different layers of the model. As shown in Fig 7 in the Appendix, this linearity is consistent across all layers, independent of model depth. Representative examples of weight linearity are provided in Fig 1c.

To verify that this linearity is a general phenomenon rather than an artifact of a specific configuration, we extend our experiments to cover diverse model sizes (from 1.5B to 8B), architec-

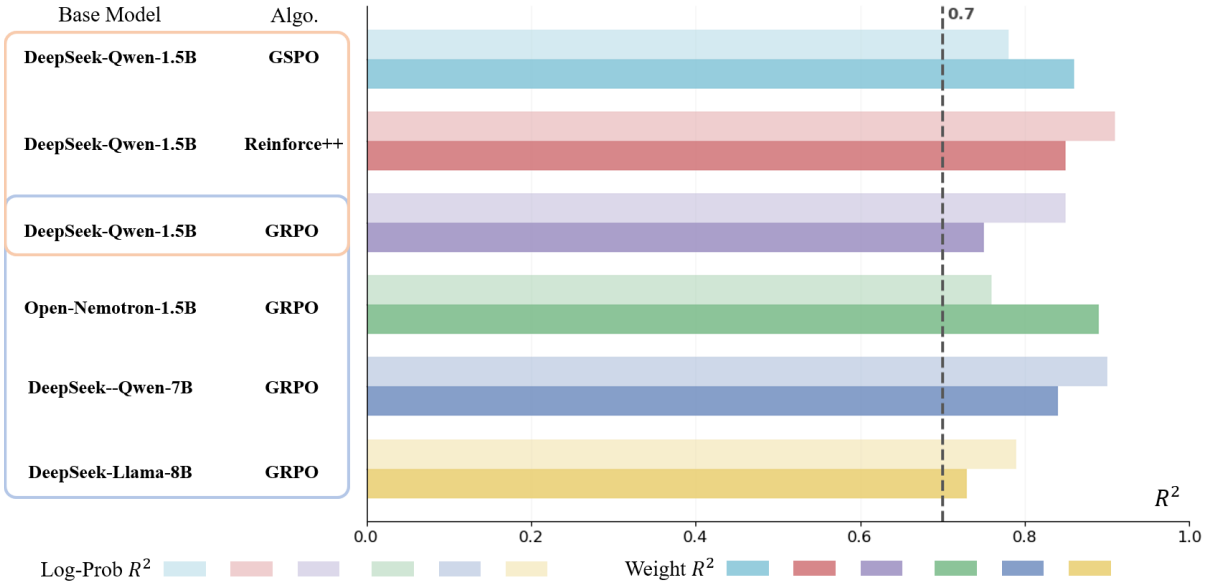


Figure 2: Linearity consistency across diverse experimental setups. The R^2 scores consistently exceed 0.7 (dashed line) across various base models (e.g., DS-Qwen, DS-Llama), scale sizes (1.5B to 8B), and training algorithms (GSPO, Reinforce++, and GRPO). The high R^2 values for both token log-probabilities and weights indicate a robust linear relationship that persists across architectural and algorithmic configurations.

tures (Qwen and Llama), training data, and RL algorithms (GRPO, Reinforce++, and GSPO). As detailed in Figure 2, we observe consistent high linearity across all settings, with the weight-level R^2 exceeding 0.7 for all combinations. For instance, scaling to the 7B parameter regime (DeepSeek-R1-Distill-Qwen-7B on Skywork-OR1-RL) or changing the architecture to Llama-8B yields similar results. Furthermore, the phenomenon persists across varied training datasets and different RL algorithms (training details are provided in Appendix A.1). These results suggest that weight linearity is an intrinsic characteristic of the RLVR for reasoning models.

3.2 Linearity in Model Outputs

In this part, we view the LLMs as a black box to analyze behavioural shifts during RLVR. Since the conditional token probabilities directly control model's generation, we focus our analysis on the evolution of log-probabilities.

Similar to analysis on weight linearity, we perform a linear fit on the token log probabilities with respect to the RL training steps. Specifically, we generate responses for AIME24 queries using the base model (64 samples per query) (evaluation details are provided in Appendix A.1), and track the log probabilities of these generated tokens across all subsequent training checkpoints.

As shown in Figure 1b, the distribution of R^2 is centered around 0.9, demonstrating that token log probabilities evolve linearly. Consistent with our weight analysis, we also verified this phenomenon in different settings. As reported in Figure 2, high linearity in log-prob is preserved across varying base models, training data, and algorithms, with token-level R^2 exceeding 0.7 for all combinations.

Notably, we observe a positive correlation between the magnitude of the update and linearity: groups with larger log-probability changes exhibit higher R^2 values, indicating that the most significant behavioral shifts occur in a strictly linear fashion. We further categorized tokens into three distinct groups (as shown in Figure 8). The first and most notable category consists of tokens characterized by both high variance and high R^2. These are largely behavioral indicators—such as reasoning connectors like 'wait', 'but', and 'therefore', and some tokens that follow them—which serve to steer the generation process; their probabilities evolve linearly, reflecting the model's steady alignment with specific response patterns. The second category, consisting of tokens with high variance but low R^2, represents a small minority that exhibits stochastic fluctuations. The final group comprises stable tokens, whose log probability rarely changes, predominantly associated with mathematical calculation components.

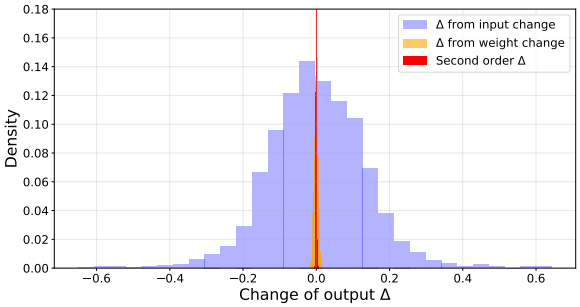


Figure 3: The source of output changes in a representative LLM layer.

3.3 Origin of Linearity

From the previous experiments, we can conclude that the strong linearity of model weights and outputs is a fundamental phenomenon in RLVR training. However, the observed linearity is unnatural given the highly non-linear structure of transformers. In this part, we analyze the origin of the linearity from a theoretical perspective. We will first analyze the relationship between weight linearity and output linearity. Then we delve into training details to find the root source of weight linearity.

Weight linearity leads to output linearity Even if model weights update linearly, it is still surprising that the intermediate and final outputs of the LLMs all exhibit strong linearity during RLVR training, given the strongly non-linear computation flow of transformers.

For the simplicity of explanation, we will randomly pick a linear layer $y = Wx$ in the MLP of a transformer layer as an example for analysis, where x, y, and W are the input, output, and weight matrix of the current layer, respectively. We will easily notice that, even though the weight matrix $W = W^0 + W't$ and input $x = x^0 + x't$ are both linear functions of training step t, $y = (W^0 + W't)(x^0 + x't) = W^0x^0 + (W'x^0 + W^0x')t + W'x't^2$ should be a quadratic instead of a linear function of t.

In fact, we find in our experiment that the quadratic term $W'x't^2$ is very small compared with the linear term $W^0h'_{i-1}t$, which will dominate the change of h_i. Fig 3 shows the contribution of the first and second order terms for the change of outputs of a linear layer in the transformer. We can see that the output change is dominated by the first-order impact of input and weight changes, while the second-order term is uniformly small among samples. Because of the low precision of BF16, in most cases, the second order term will have no impact on the outputs at all.

We can also see that the change in output y mainly results from the change of the input x, which accumulates small changes of weights in previous layers. For attention and embedding layers, we can derive the same conclusion with roughly the same analysis. As a result, for the same input of the transformer, the linearity will propagate to activations at high layers and even the output logits, when the weights change linearly.

The source of weight linearity We believe that the Adam optimizer is one of the key reasons for the weight linearity of RLVR training. Unlike stochastic gradient descent (SGD), the stability of the gradient, rather than its absolute value that determines the per-step weight update during training. In RLVR, because of the small learning rate (usually < 1e-5), and relatively large batch size (usually >128 (mini batch size)× 8 (rollout number)), the distribution of gradients will tend to be stable during training. As a result, the speed of weight update will remain stable, leading to the linearity of weights.

4 Accelerating RLVR with linearities

The linearities of RLVR training indicate that the weights and output at a certain step can be largely predicted by its training trajectory at earlier steps. As a result, we can speed up RLVR training by replacing standard training of some steps with linear extrapolation. In the following, we will describe direct extrapolation from two perspectives: Weight Extrapolation and Logits Extrapolation. We then introduce RL-Extra, an interactive training scheme that interleaves extrapolation with RL updates.

4.1 Experimental Setup

We utilize the DeepScaleR-Preview dataset (Luo et al.) to post-train a DeepSeek-R1-Distilled-Qwen-1.5B model (DeepSeek-AI et al., 2025) via reinforcement learning. To rigorously evaluate our method, we employ four widely used benchmarks: AIME-24/25 (Art of Problem Solving, 2025), MATH-500 (Lightman et al., 2024), and LiveCodeBench (v5, Oct 2024 – Feb 2025) (Jain et al., 2025). These benchmarks span mathematical reasoning and programming tasks, providing a comprehensive assessment of our model's capabilities. Detailed training configurations and metrics are provided in Appendix A.1.

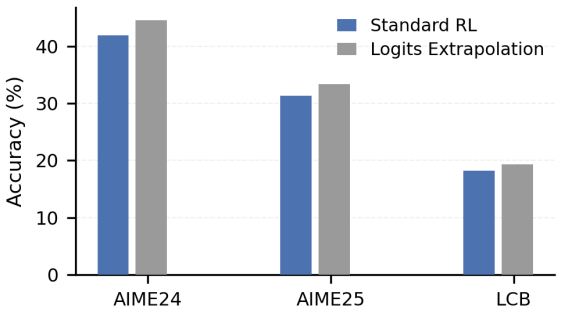


Figure 4: Accuracy comparison on AIME and LCB benchmarks. Logit Extrapolation yields consistent improvements over standard RL across all evaluated settings.

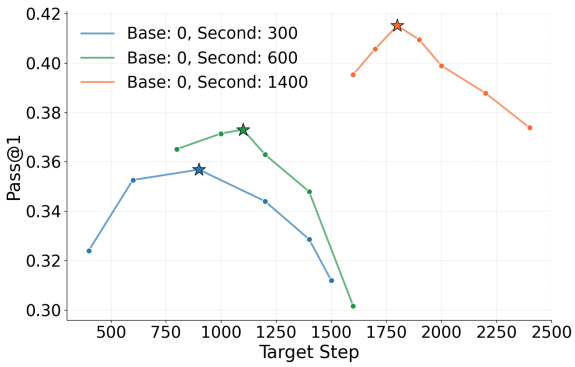


Figure 5: Weight Extrapolation performance on AIME24 across different target steps.

4.2 Direct Extrapolation

We first investigate direct extrapolation on output logits and model weights. Namely, given model checkpoints at two different time steps, we can directly predict the model at a future step by linear extrapolation.

4.2.1 Logits Extrapolation

We first investigate naive extrapolation on logits. Specifically, we approximate the policy distribution of a future checkpoint (denoted as step t') by leveraging the logits from two preceding checkpoints, $t_0 < t_1$. Formally, given an input sequence $\mathbf{x}$, let $\mathbf{l}_k$ denote the logits vector produced by the model at step k. We project the logits for the target step, $\mathbf{l}_{t+1}$, via linear extrapolation:

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

where $\alpha = \frac{t'-t_0}{t_1-t_0} > 1$ is a coefficient controlling the magnitude of the extrapolation. This formulation enables the simulation of the policy's sampling trajectory at a future state solely through vector arithmetic on logits, without the computational overhead of explicit gradient updates.

Figure 4 demonstrates the avg@k performance of Logits Extrapolation on AIME24/25 and LiveCodeBench. Specifically, This superior performance on mathematical and coding benchmarks demonstrates that Logits Extrapolation can surpass the performance boundaries of standard RL. We attribute this performance gain to the method's ability to mitigate late-stage training instability. During the RLVR process, prolonged training often leads to entropy collapse and overfitting, causing the actual model trajectory to deviate from the optimal generalization path. Logits Extrapolation captures the stable optimization direction established in earlier steps and projects it forward, thereby preserving the linearity of improvement while avoiding the degradation associated with excessive gradient steps.

4.2.2 Weight Extrapolation

In this part, we first introduce Weight Extrapolation, which directly predicts model weights at t', from checkpoints at previous time steps t_0 and t_1. Formally, let $\mathbf{W}_k$ denote the model weights (parameters) at optimization step k. By utilizing the optimization trajectory observed from two historical checkpoints, specifically steps t_0 and t_1, we linearly project the weights to estimate the model configuration at a future step t':

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

where $\beta > 1$ is the extrapolation coefficient for the parameter space. This projected weight configuration $\mathbf{W}_{t'}$ constitutes a virtual lookahead model.

As shown in Figure 5, we fix t_0 and t_1 and vary the extrapolation step size $(t' - t_1)$, plotting the performance of Weight Extrapolation on AIME24 as a function of the equivalent extrapolation step. Specifically, starting from three different t_1, the performance of Weight Extrapolation exhibits an inverted U-shaped as a function of t'. For example, the blue line fixes $t_0, t_1 = 0, 300$, and increases t' from 400. The best performance is achieved when $t' = 900$, approaching 0.36. As the extrapolation t' further increases, the performance of Weight Extrapolation begins to decline. This indicates that there is a limit for direct weight extrapolation, as models may still need to accumulate subtle deviations from original linear trajectories to partially modify their updating directions.

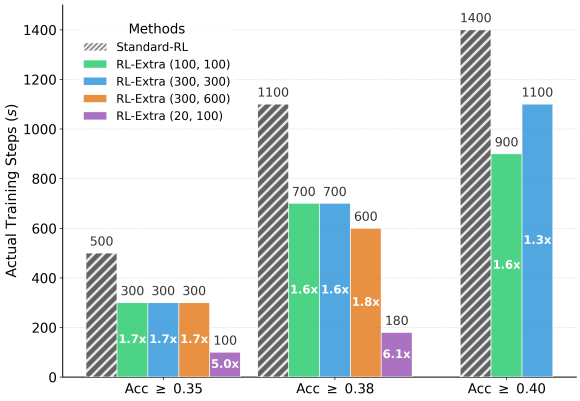


Figure 6: Comparison of actual training steps required to reach target accuracy on AIME24.

4.3 RL-Extra

Taking the locality of weight extrapolation into consideration, we propose RL-Extra, a paradigm that interleaves actual training with weight extrapolation. By periodically grounding the model with gradient updates, we can correct the optimization trajectory, thereby enabling a larger extrapolation training ratio without divergence.

Formally, RL-Extra operates in cycles of period $C = m + n$. Each cycle begins with m steps of standard gradient-based optimization to align with the true reward signal, followed by n steps of gradient-free extrapolation to accelerate progress. Let k denote the current global step. The update rule for the model parameters $\mathbf{W}_{k+1}$ is formalized as follows:

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

where $\mathcal{L}_{\text{RL}}$ denotes the RL objective function and η is the learning rate. During the first phase, the model updates via standard gradient descent. In the subsequent extrapolation phase, the model continues to evolve along the established trajectory solely through linear projection, thereby reducing the computational overhead while maintaining optimization momentum.

As presented in Table 3, we conduct a comparative analysis between RL-Extra and standard RL training (GRPO) under different training budgets, measured by actual training steps $s \in \{200, 400, 800, 1200\}$. We can see that RL-Extra consistently outperforms the standard RL baseline across all benchmarks under all budget constraints (specific configurations for each budget are detailed in Appendix A.1). We attribute these gains to the inherent linearity in the weight space during RL training, which allows the estimated optimization trajectory to accurately project future states via weight extrapolation. Crucially, since this process requires no additional gradient updates, it incurs zero additional GPU training costs, effectively offering a "free lunch" for performance improvement.

Figure 6 breaks down RL-Extra under different hyperparameter settings. Here, setting (m, n) denotes alternating between m RL steps and n extrapolation steps; we report the number of *actual* RL training steps required by standard RL to reach the same AIME24 accuracy (0.35, 0.38, and 0.40 in three matched-performance comparisons). For example, to achieve the same level of SOTA performance of standard RL at 0.40, RL-Extra $(100, 100)$ requires only 900 RL steps, corresponding to a $1.6\times$ speedup.

We also evaluate a more aggressive schedule, RL-Extra $(20, 100)$ (20 RL steps followed by 100 extrapolation steps; a $5\times$ disparity). Despite its extreme ratio of extrapolation and actual training, this configuration attains > 0.38 AIME24 accuracy with only 180 RL steps, matching the performance of standard RL trained for 1100 steps and yielding a $6.1\times$ speedup.

Overall, these results show that RL-Extra can make more efficient use of information from each training step to speed up RLVR training. We attribute these gains to the approximate linearity of weight-space dynamics during RL training, which allows weight extrapolation to accurately project future points along the optimization trajectory.

5 Conclusion

In this paper, we introduce the strong, universal linear trends in model weights and outputs across RL training steps. Leveraging this, we propose Direct Extrapolation (Weight/Logits) and RL-Extra, achieving up to a **3%** gain and a **6.1**$\times$ wall-clock speedup. Our future work focuses on two key directions. First, motivated by the strong non-linearity observed at the point of entropy collapse, we aim to further investigate the root causes of this phenomenon. Second, we will examine the specific dynamics of the RLVR process, where gradient accumulation is used to perform large parameter updates after aggregating gradients over many steps, rather than frequent incremental updates.

Limitations

First, regarding model scale and architecture, our experiments were primarily conducted on dense models with fewer than 30 billion parameters. We have not yet verified whether the observed linearity generalizes to ultra-large-scale models (e.g., >30B parameters) or sparse architectures such as Mixture-of-Experts (MoE). Second, our experimental setting for Reinforcement Learning (RL) did not encompass complex multi-turn interactions; thus, the stability of linearity during multi-turn RL optimization remains to be explored. Finally, this work focuses on empirical analysis in a research environment and has not yet been validated in large-scale industrial deployment scenarios.

References

Art of Problem Solving. 2025. AIME problems and solutions. https://artofproblemsolving.com/wiki/index.php/AIME_Problems_and_Solutions.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, and 1 others. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*.

Howard Chen, Noam Razin, Karthik Narasimhan, and Danqi Chen. 2025. Retaining by doing: The role of on-policy data in mitigating forgetting. *Preprint*, arXiv:2510.18874.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 181 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Hugging Face. 2024. Open-r1: An open initiative to replicate deepseek-r1. Accessed: January 6, 2026.

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

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *Preprint*, arXiv:2106.09685.

Jian Hu, Jason Klein Liu, Haotian Xu, and Wei Shen. 2025a. Reinforce++: Stabilizing critic-free policy optimization with global advantage normalization. *Preprint*, arXiv:2501.03262.

Jian Hu, Mingjie Liu, Ximing Lu, Fang Wu, Zaid Harchaoui, Shizhe Diao, Yejin Choi, Pavlo Molchanov, Jun Yang, Jan Kautz, and Yi Dong. 2025b. Brorl: Scaling reinforcement learning via broadened exploration. In *arXiv preprint arXiv:2510.01180*. Accessed: January 6, 2026.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, and 1 others. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. 2025. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Aayush Karan and Yilun Du. 2025. Reasoning with sampling: Your base model is smarter than you think. *Preprint*, arXiv:2510.14901.

David Joseph Kopiczko, Tijmen Blankevoort, and Yuki M Asano. 2023. Vera: Vector-based random matrix adaptation for large language models. *Preprint*, arXiv:2310.11454.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, and 1 others. 2024. Tulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*.

Hung Le, Yue Wang, Akhilesh Deepak Gotmare, Silvio Savarese, and Steven C. H. Hoi. 2022. Coderl: Mastering code generation through pretrained models and deep reinforcement learning. *Preprint*, arXiv:2207.01780.

Yanda Li, Shiyang Li, Chao Li, Yifeng Yu, Yizhou Zhang, and Yang Gao. 2025. How rl after next-token prediction facilitates learning. *Preprint*, arXiv:2510.11495.

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

Shih-yang Liu, Chien-Yi Wang, Tzu-Chieh Yin, Jhih-Jie Jhang, Yung-Chen Wang, Mau-Tsu Chen, and Hung-yi Wang. 2024a. Dora: Weight-decomposed low-rank adaptation. *Preprint*, arXiv:2402.09353.

Zichang Liu, Liang Wang, Zhen Cao, and 1 others. 2024b. Milora: Minor singular components initialization for low-rank adaptation. *Preprint*, arXiv:2405.18415.

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y. Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Li Erran Li, Raluca Ada Popa, and Ion Stoica. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl. https://notion.so/19681902c1468005bed8ca303013a4e2.

Fanxu Meng, Zhaohui Wang, Mingxuan Zhang, Han Li, Jian Jiang, Lei Zhang, Chen Yang, Xuyi Sun, Wei Chen, Xu Jiang, and 1 others. 2024. Pissa: Principal singular values and singular vectors adaptation of large language models. *Preprint*, arXiv:2404.07882.

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

Youssef Mroueh. 2025. Reinforcement learning with verifiable rewards: Grpo's effective loss, dynamics, and success amplification. *arXiv preprint arXiv:2503.06639*.

Sagnik Mukherjee, Lifan Yuan, Dilek Hakkani-Tur, and Hao Peng. 2025. Reinforcement learning finetunes small subnetworks in large language models. *arXiv preprint arXiv:2505.11711*.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. *Preprint*, arXiv:2203.02155.

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

Idan Shenfeld, Jyothish Pari, and Pulkit Agrawal. 2025. Rl's razor: Why online reinforcement learning forgets less. *Preprint*, arXiv:2509.04259.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. 2022. Solving math word problems with process- and outcome-based feedback. *Preprint*, arXiv:2211.14275.

Xinyi Wang, Jason Wei, Dale Schuurmans, and 1 others. 2022. Self-consistency improves chain of thought reasoning in language models. In *ICLR*.

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. 2025. Reinforcement learning for reasoning in large language models with one training example. *Preprint*, arXiv:2504.20571.

Yue Wang, Hung Le, Akhilesh Gotmare, Nghi Bui, Junnan Li, and Steven Hoi. 2023. Codet5+: Open code large language models for code understanding and generation. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pages 1069–1088.

Fang Wu, Weihao Xuan, Ximing Lu, Mingjie Liu, Yi Dong, Zaid Harchaoui, and Yejin Choi. 2025. The invisible leash: Why rlvr may or may not escape its origin. *Preprint*, arXiv:2507.14843.

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

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Shiji Song, and Gao Huang. 2025. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? *arXiv preprint arXiv:2504.13837*.

Qingru Zhang, Min Chen, Alexander Bukharin, Daniel Khashabi, Dan Roth, Beiye Chen, and Diyi Yang. 2023a. Adalora: Adaptive budget allocation for parameter-efficient fine-tuning. *Preprint*, arXiv:2303.10512.

Rui Zhang, Lefei Liu, Peng Wang, and Lizhen Qiu. 2023b. Miss: Mixture of sub-spaces for parameter-efficient fine-tuning. *Preprint*, arXiv:2310.18168.

Yifan Zhang, Yifan Zhang, Mingjie Liu, Haoran Wang, Ximing Lu, and Yi Dong. 2025. Logic-rl: Unveiling the emergence of complex reasoning in large language models via reinforcement learning. *Preprint*, arXiv:2502.14768.

Wayne Xin Zhao, Kun Liu, Yilei Zhang, and 1 others. 2025. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for LLM reasoning. *Preprint*, arXiv:2506.01939.

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. 2025. Group sequence policy optimization. *Preprint*, arXiv:2507.18071.

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. 2025. The path not taken: Rlvr provably learns off the principals. *Preprint*, arXiv:2511.08567.

A Appendix

A.1 Experimental Setup

Hyperparameter	Value
KL Loss	No
Entropy Regularization	No
Global Batch Size	[128, 256, 512]
PPO Mini-batch Size	[64, 256]
Max Response Length	16K
Learning Rate	1×10^{-6} (Constant)
Clip Ratio Range	$[0.8, 1.28]$
Temperature	1.0
Rollout (N)	16

Table 1: **Hyperparameter settings.** These settings are applied consistently across GRPO, GSPO, and Reinforce++.

Training Details The training hyperparameters are shown in the Table 1. We adapt our training codebase from Verl and follow the training recipe of 3 different RL algorithms, including GRPO, GSPO, and Reinforce++. For Reinforce++, critic optim learning rate is $9e^{-6}$.

Evaluation Setup We evaluate models on 4 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.

For RL-Extra, the specific configurations (parameters m and n) selected for each training budget are detailed in Table 2.

A.2 Additional Results

Weight Linearity 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.

Table 2: **RL-extra Hyperparameter Configurations.** This table details the specific values for parameters m and n used in the RL-extra experiments corresponding to each fixed training budget reported in Table 3.

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

Output Linearity Figure 8 shows examples of different dynamics of log probabilities. Fig. 9 shows the R^2 distribution of all activations in different layers of the transformer. Similar to the conclusion on log probabilities, we can conclude that all intermediate layers except for the first one exhibit strong linearity against training steps.

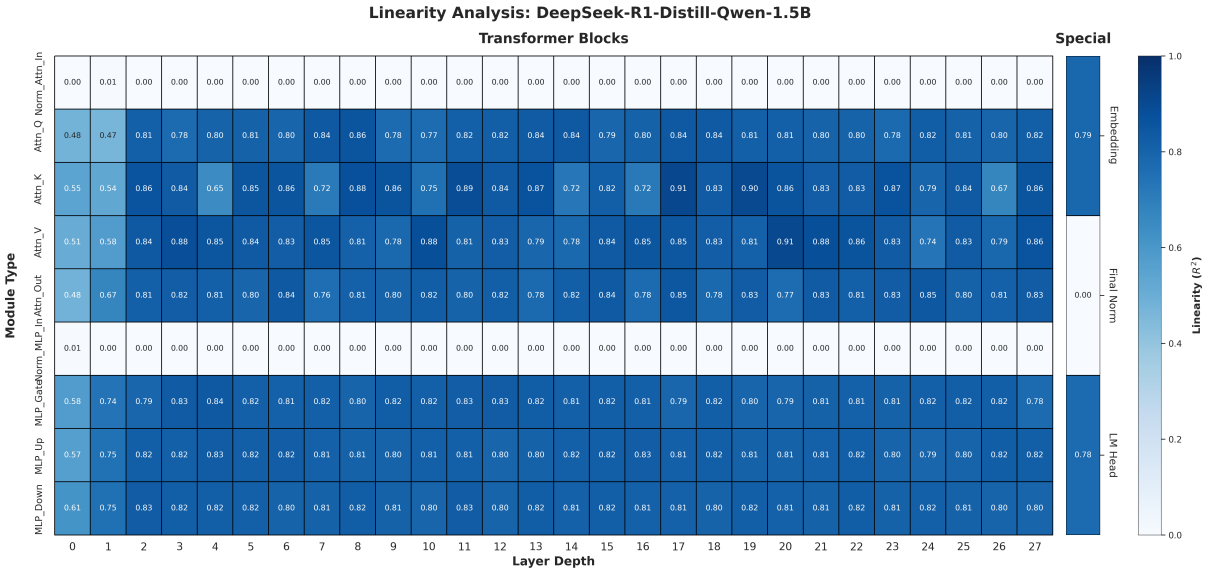


Figure 7: Evolution of weight linearity across model layers during RLVR training. The figure displays the average R^2 from a linear fit of the weights at each layer. Note that due to the small number of parameters in Layer Normalization layers, no filtering was applied to them.

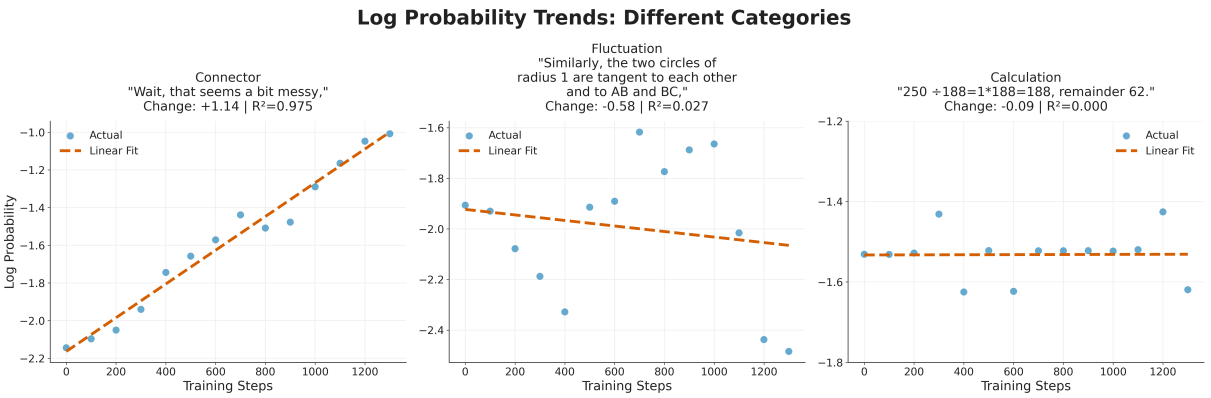


Figure 8: Case study of tokens log-probability dynamics. The left panel shows tokens acting as logical connectors, characterized by significant log-probability changes and high R^2 values; the middle panel displays tokens with large variation in log-probability but low R^2, where the probability fluctuates irregularly; the right panel depicts tokens with smaller log-probability variations, which are mostly components of mathematical calculations.

Table 3: **Performance Comparison under Fixed Training Budgets.** We evaluate RL-Extra against the GRPO baseline across AIME24, AIME25, MATH500, and LiveCodeBench. When the training budget is fixed at specific actual training steps (s), our method consistently achieves higher performance than the baseline.

Training Steps (s)	Method	AIME24 (Avg@64)	AIME25 (Avg@64)	MATH500 (Avg@64)	LCB (Pass@4)
200	Standard RL	0.3172	0.2536	0.8421	**0.2714**
	RL-Extra	**0.3318**	**0.2979**	**0.8611**	0.2619
400	Standard RL	0.3391	0.2682	0.8525	0.2810
	RL-Extra	**0.3672**	**0.3005**	**0.8658**	**0.2905**
800	Standard RL	0.3490	0.2932	0.8664	0.2821
	RL-Extra	**0.3984**	**0.3094**	**0.8667**	**0.2905**
1200	Standard RL	0.3828	0.2995	0.8658	**0.2857**
	RL-Extra	**0.4120**	**0.3120**	**0.8731**	0.2762

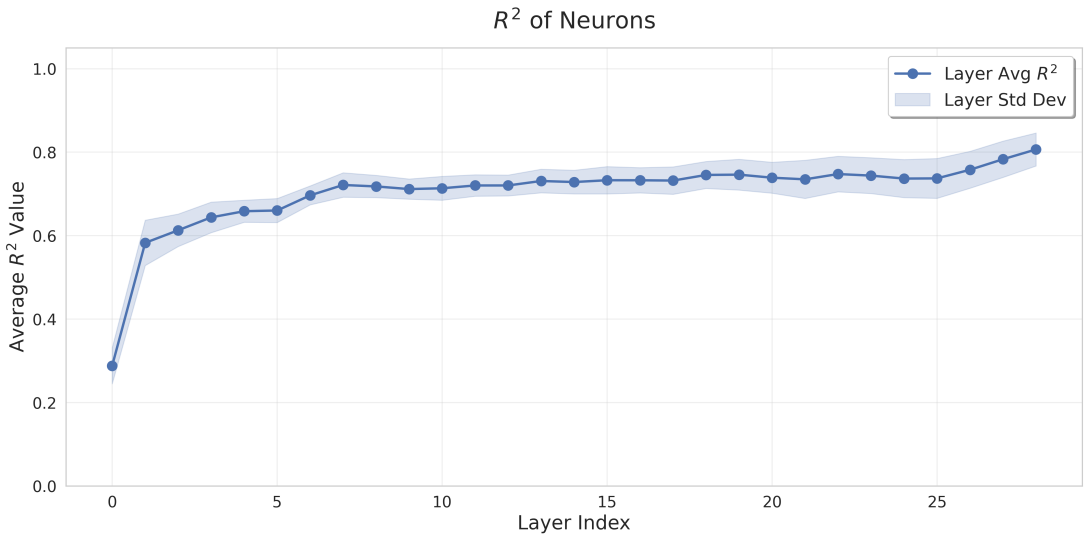


Figure 9: The R^2 distributions of activations across different layers.