LEARNING RATE TRANSFER FOR REINFORCEMENT LEARNING POST-TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning (RL) post-training of language models can be unstable and sensitive to hyperparameter choices, requiring extensive tuning that is particularly costly for larger models. We investigate whether learning rates selected on a smaller model can transfer to wider and deeper models during RL post-training. We show the Group Relative Policy Optimization (GRPO) objective is compatible with the CompleteP feature-learning scaling prescription. Across five verifiable-reward tasks and models with up to approximately one billion parameters, the scaled parameterization supports more reliable learning rate transfer than the standard parameterization, with a wider shared range of high performing rates across widths. These results support tuning on smaller models as a practical approach to reducing the hyperparameter search cost of RL post-training.

1 INTRODUCTION

As language models have grown enormously in size, the cost of training, and therefore the cost of repeating training runs for hyperparameter tuning, has become substantial (Hoffmann et al., 2022). Hyperparameter transfer offers an appealing method to reduce this cost: we tune on a smaller model and use the resulting hyperparameters as an estimate of well-performing hyperparameters for a larger model. Yang et al. (2022) introduced this method and showed that transfer is possible when the two models are related by an appropriate scaling prescription. This method allows for a much cheaper search for hyperparameters since the model used for tuning can be kept significantly smaller, while the larger model can still yield improved performance.

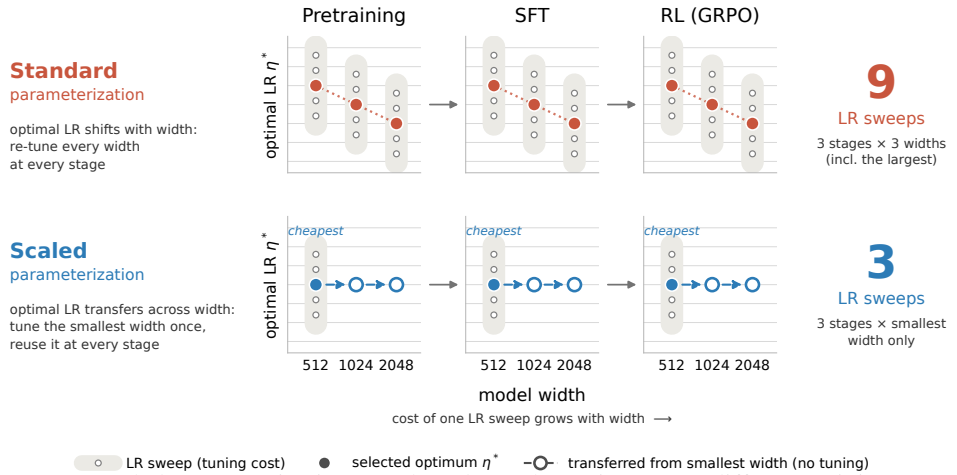


Figure 1: Scaled parameterization greatly reduces the tuning across different training stages.

As the development of language models has progressed, reinforcement learning (RL) has become an increasingly important and computationally expensive part of training, as it requires repeated rollouts to evaluate and update the policy (Ouyang et al., 2022; Shao et al., 2024; DeepSeek-AI, 2025). Here, the challenge of finding well-performing hyperparameters is even more important, as RL post-training is known to be particularly unstable and sensitive to hyperparameters (Ouyang

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.

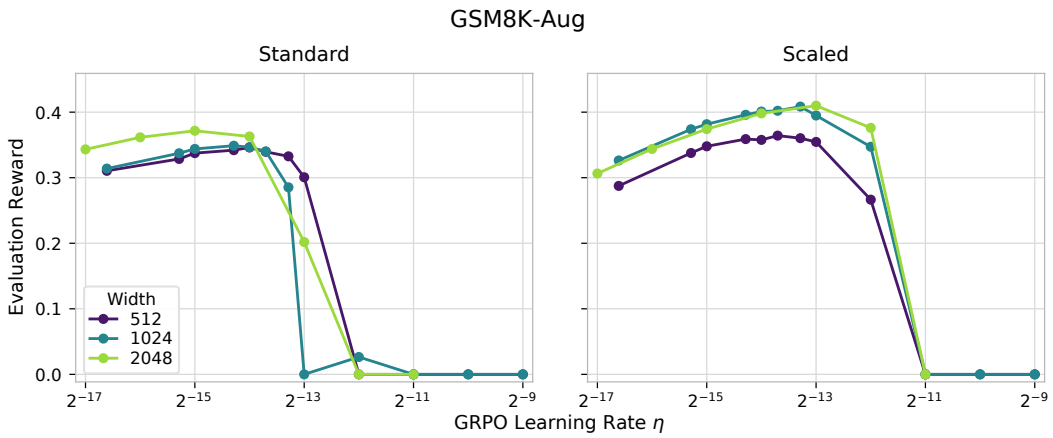


Figure 2: Scaled parameterization (right) exhibits a much more stable learning rate transfer across model widths in RL post-training on GSM8K-Aug compared to standard parameterization (left).

et al., 2022; Yu et al., 2025). Indeed, in our experiments, an inappropriate choice of learning rate can cause the reward to collapse to zero. Given the multi-stage nature of the training pipeline, it is unclear whether learning rate transfer can be applied to post-training, and if so, what scaling prescriptions are appropriate. Therefore, our main question is

Can learning rates tuned on smaller models transfer to wider and deeper models during RL post-training?

A priori, there is no reason to expect that learning rate transfer should hold during RL post-training, as the training is in a new stage with different objectives and data. In fact, the infinite-width feature-learning theory underlying these scaling prescriptions is typically limited to a fixed finite training horizon (Yang & Hu, 2021), whereas RL post-training begins only after extensive pretraining and supervised fine-tuning, when finite-width corrections are likely to be significant (Bordelon & Pehlevan, 2024; Vyas et al., 2023). Indeed, at the beginning of RL, selected-matrix maximal-update measurements are not uniformly stable across width, even under the scaled parameterization. Therefore, the existing maximal-update theory does not by itself predict whether or not learning rate transfer should hold during post-training. Resolving this question requires both an analysis of how the RL objective scales with width and depth, and a characterization of training dynamics in multiple extended stages of training. We discuss more related works in Appendix B.

Our main contributions are as follows:

- We examine the maximal-update diagnostic at the start of RL and identify limits of its interpretation after multistage training. We also estimate selected-matrix positive-curvature boundaries for the base learning rate. On GSM8K these boundaries are comparatively more stable under scaling, while WikiSQL shows that the pattern is not universal.
- We analyze the GRPO objective and identify conditions under which it is compatible with the feature-learning width and depth scaling prescription of CompleteP (Dey et al., 2025).
- Across five verifiable-reward tasks and models up to one billion parameters (width 2048 and depth 8), we empirically verify that learning rates transfer across width more reliably under feature-learning scaling than under standard parameterization (see Table 2 and Appendix E). On GSM8K we also observe depth transfer (Figure 8).

Together, these results establish that feature learning scaling remains an important prescription for large-scale language model post-training. Figure 2 illustrates the learning-rate range available across model widths during RL post-training.

2 Setup and Scaling Framework

We consider a decoder-only Transformer based on the Llama 3.2 architecture (Meta, 2024). Let n denote the width, d depth or number of Transformer blocks, n_0 its vocabulary size, and n_h its

attention-head dimension. The number of query heads is $n_{\text{head}} = n/n_h$. We assume that n is divisible by n_h.

For a sequence $x = (x^1, \dots, x^m)$, let $h_\ell^\alpha \in \mathbb{R}^n$ denote the representation of token α entering block ℓ, and write $h_\ell = [h_\ell^1, \dots, h_\ell^m] \in \mathbb{R}^{n\times m}$ for the matrix form. The embedding matrix $W_0 \in \mathbb{R}^{n\times n_0}$ gives $h_1^\alpha = W_0 e^\alpha$, where e^α is the one-hot vector associated with token x^α. For $h, \gamma \in \mathbb{R}^n$, denote the RMSNorm output (Zhang & Sennrich, 2019) with gain γ by $h^{\text{RN}} := \gamma \odot h \left(\frac{1}{n}\|h\|_2^2 + \varepsilon_{\text{rms}}\right)^{-1/2}$ where $\varepsilon_{\text{rms}} > 0$.

For $\ell = 1, \dots, d$, the attention weight matrices are $W_\ell^q, W_\ell^k, W_\ell^v, W_\ell^o \in \mathbb{R}^{n\times n}$. The gated MLP block (Shazeer, 2020) has intermediate width $4n$, with $W_\ell^1, W_\ell^3 \in \mathbb{R}^{4n\times n}$ and $W_\ell^2 \in \mathbb{R}^{n\times 4n}$. We use the SiLU activation, $\varphi(h) = \frac{h}{1+\exp(-h)}$ for scalar h (Elfwing et al., 2018).

The architecture is summarized by the following recursion for $\ell \in [d]$

$$\begin{aligned} \bar{h}_{\ell+1} &= h_\ell + \frac{1}{d} W_\ell^o \left[(W_\ell^v h_\ell^{\text{RN}})_j A_\ell^j \right]_{j=1}^{n_{\text{head}}}, \quad A_\ell^j = \text{Softmax}\left(\frac{1}{n_h} (W_\ell^k h_\ell^{\text{RN}})_j^\top (W_\ell^q h_\ell^{\text{RN}})_j \right), \\ h_{\ell+1} &= \bar{h}_{\ell+1} + \frac{1}{d} W_\ell^2 \left(\varphi\left(W_\ell^1 \bar{h}_{\ell+1}^{\text{RN}}\right) \odot W_\ell^3 \bar{h}_{\ell+1}^{\text{RN}} \right), \end{aligned} \tag{1}$$

and the output is $f(x) = \frac{n_b}{n} W_{d+1} h_{d+1}^{\text{RN}}$.

Here the subscript $j \in [n_{\text{head}}]$ of a matrix (Wh) selects its j-th block of n_h rows to capture the j-th attention head. The activation φ acts entry-wise, and Softmax is column-wise with causal masking, and the square brackets stack the head outputs vertically. In block ℓ, h_ℓ^{RN} is normalized column-wise with trainable gain γ_ℓ, while $\bar{h}_{\ell+1}^{\text{RN}}$ uses $\widetilde{\gamma}_\ell$. The final normalized state h_{d+1}^{RN} uses γ_{d+1}. All three gain vectors have width n, and n_b is a constant base width.

We suppress rotary position embeddings (Su et al., 2021) and the key–value head-sharing pattern (Ainslie et al., 2023) in the recursion, as neither is used in the scaling argument.

The main differences between the standard and scaled parameterizations are the residual and output multipliers, as well as the initialization and optimizer scalings. We summarize the details in Table 1.

Here $a \in \{q, k, v, o, 1, 2, 3\}$ is the superscript that denote the type of weight matrix, $1 \le \ell \le d$ is the layer index, n_b is the base width, and T is the training horizon in optimizer steps. The scaling is equivalent to CompleteP (Dey et al., 2025). The scaled weight-decay factors use the optimizer-step horizon T, motivated by the training-horizon analysis of Wang & Aitchison (2025). The base optimizer values η_b, λ_0, and ϵ are specified for each parameterization and training stage. The scaled weight-decay entries assume that the configured base decay is λ_0/T; the optimizer does not divide by T internally.

Category	Variable	Standard	Scaled
Multiplier	Residual Branch $h_{\ell+1}, \bar{h}_{\ell+1}$	1	$1/d$
	Output f	1	n_b/n
Init. Variance	W_0	σ^2	σ^2
	W_ℓ^a	σ^2	$\sigma^2 n_b/n$
	W_{d+1}	σ^2	σ^2
Learning Rate η	W_0, W_{d+1}	η_b	η_b
	W_ℓ^a	η_b	$\eta_b n_b/n$
	γ	η_b	η_b
Weight Decay λ	W_0	λ_0	0
	W_{d+1}	λ_0	λ_0/T
	W_ℓ^a	λ_0	$\lambda_0 n/(n_b T)$
	γ	0	0
AdamW ϵ	$W_0, W_{d+1}, \gamma_{d+1}$	ϵ	$\epsilon n_b/n$
	$W_\ell^a, \gamma_\ell, \widetilde{\gamma}_\ell$	ϵ	$\epsilon n_b/(nd)$

Table 1: Scaling prescriptions for the standard and scaled parameterizations. The multiplier for standard parameterization is removing $1/d$ in the residual branch and n_b/n in the output from (1).

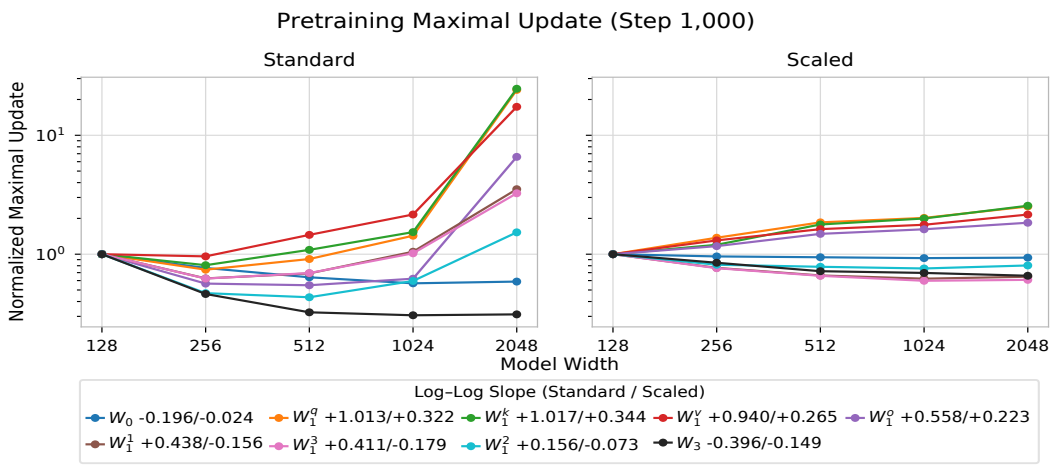


Figure 3: The scaled parameterization's (right) maximal update effects vary substantially less with width after 1,000 steps of pretraining, while the standard parameterization (left) shows much more variation. Here we measure a normalized version of the maximal update effect $\widehat{M}_W(n) = M_W(n)/M_W(128)$ from (2).

We consider a three stage training pipeline: pretraining, supervised fine-tuning (SFT), and RL post-training. Each model is initialized randomly and pretrained with an autoregressive next-token prediction objective. For each downstream task, supervised fine-tuning starts from a pretrained checkpoint and minimizes cross-entropy on response tokens. Finally, we start RL post-training from the task specific fine tuned checkpoint, and optimizes verifiable rewards with Group Relative Policy Optimization (GRPO) (Shao et al., 2024).

3 Maximal Update Across Training Stages

First introduced by Yang & Hu (2021), the maximal-update desideratum is a scale requirement on weight update's contribution towards neuron changes, which the authors formulate as a necessary condition for feature learning in the infinite-width limit. Consequently, Yang et al. (2022) further argues this as a requirement for hyperparameter transfer. The notion has since been refined further by Dey et al. (2025), where the authors provide several desired conditions for transfer, including a depth adjusted version maximal update, which we will state as follows.

Desideratum 3.1 (Maximal Update) *Let $\Delta W = W_+ - W$ be the weight update of a parameter group W, and we let $\Delta_W h = h(W_+) - h(W)$ be the change in the neurons h induced by the weight update of W alone. We say the parameterization satisfies* ***maximal update*** *if*

$$M_{W_\ell^a}(n) := \|\Delta_{W_\ell^a} h_{\ell+1}\|_{\text{rms}} = \Theta(d^{-1/2}), \quad \forall \ell \in [d], a \in \{q, k, v, o, 1, 2, 3\}, \tag{2}$$

where the $\Theta(\cdot)$ is with respect to $d, n \to \infty$ and all other scales remain finite.

Beyond being just a theoretically desired property in order to derive an infinite-width limit (Bordelon & Pehlevan, 2022), this also yields a practical diagnostic for hyperparameter transfer. In particular, if we compare the maximal update measurements by increasing width at fixed depth, we should expect to see a nearly flat curve consistent with the desideratum's predicted width dependence. This is a practical check that the implementation approximately realizes the intended width scaling.

For example, we can observe from Figure 3 that after 1,000 steps of pretraining, the maximal update measurement is substantially flatter for the scaled parameterization than for the standard parameterization.

However, the analogous probe at the start of RL post-training gives a less uniform picture Figure 4. Starting from the task-specific GSM8K-Aug SFT checkpoints, the measurements now vary considerably across weight groups, but neither parameterization shows a distinctive pattern of width

stability or instability. As a result, this diagnostic at the early RL stage does not give a clear verdict on maximal update or learning rate transfer.

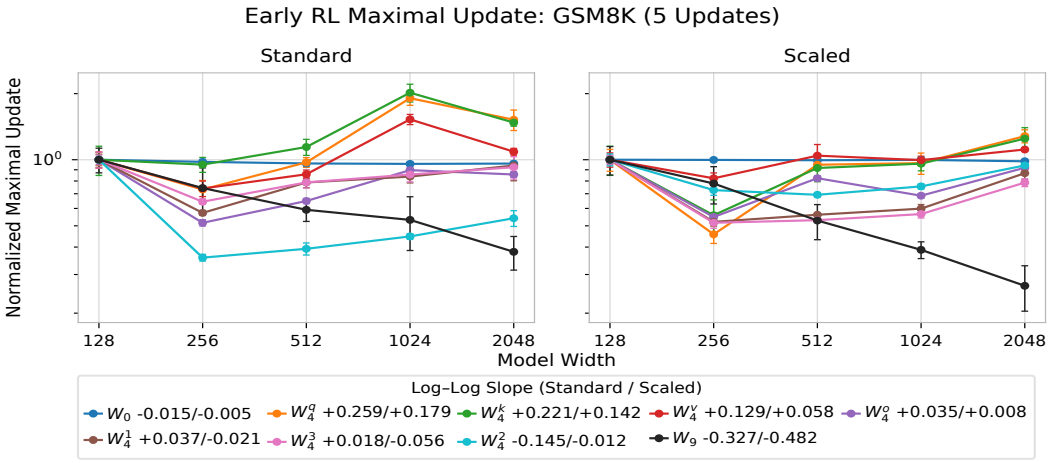


Figure 4: After five GRPO updates from the depth-eight GSM8K-Aug SFT checkpoints, neither the standard (left) nor scaled (right) parameterization shows a uniform pattern of width dependence across parameter groups. Here we are measuring a normalized version of the maximal update effect $\widehat{M}_W(n) = M_W(n)/M_W(128)$ from (2).

There are several reasons why the early RL measurement of maximal update may be inconclusive. Firstly, the infinite-width analyses above begin with random initialization, whereas RL training starts only after a pretraining and SFT stage. The multi-stage training pipeline may play an important role in the theory of training dynamics.

Secondly, the infinite width analyses motivating this diagnostic do not necessarily approximate a finite width training trajectory uniformly well over long training horizons. Bordelon & Pehlevan (2024) show that finite width corrections can significantly alter training trajectories, so the infinite-width limits may not be a good approximation in later stages of training.

Finally, comparing maximal update measurements after a pretraining stage is confounded by the fact that the pretraining horizon is not uniform across widths. The commonly accepted compute optimality rule of 20 tokens per parameter (Hoffmann et al., 2022) gives different training horizons across widths and depths, and therefore the maximal update measurement at the end of pretraining is not a pure width scaling measurement.

These limitations leave open whether the scaling prescription remains useful for RL post-training. We will next examine what we can still understand about the scaling of training dynamics that may still contribute towards learning rate transfer.

4 RL Objective Scaling and Sharpness

The failure of maximal update measurements does not immediately imply that the scaling prescriptions will fail to transfer learning rates. In this section, we explore how the GRPO objective preserves the intended scale with respect to width and depth, and secondly a curvature measurement that is more directly related to training dynamics despite the issues we discussed earlier.

We consider the token-averaged, optionally KL-regularized GRPO objective S_β defined in (9). Its rollout and clipping conventions, together with the proof, are given in Appendix C. Here $\nabla_f S_\beta$ collects the derivatives with respect to the logits at all valid response-token positions. Our experiments use $\beta = 0$.

Proposition 4.1 (GRPO Logit-Gradient Scaling) *Under the scaled parameterization in Section 2, if the reward gradient is non-vanishing after clipping, then*

$$\|\nabla_f S_\beta\|_1 = \Theta(1)\,, \quad \text{as } n, d \to \infty\,. \tag{3}$$

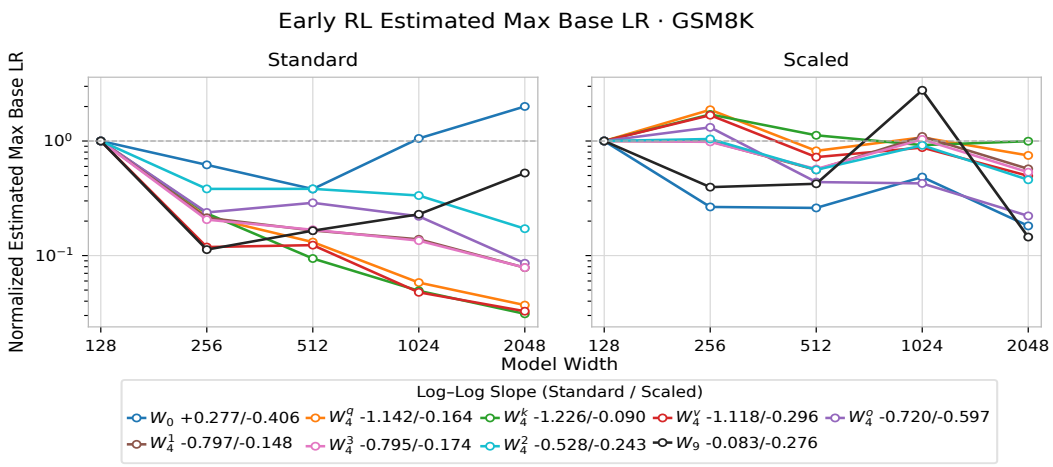


Figure 5: Scaled parameterization (right) exhibits a more stable estimated base learning-rate boundary across widths during RL post-training than the standard parameterization (left).

Indeed, this is a necessary condition for the scaling to yield learning rate transfer, as the logit gradients depending on width and depth would require additional corrections in the scaling. The proof in Appendix C follows the importance-ratio and log-policy-gradient calculation, with the clipping cases and token normalization made explicit. Thus the RL objective introduces no new width or depth factor into the loss-to-logit derivative.

There is a separate reason to expect transfer to remain possible even when the maximal-update diagnostic varies with scale. Noci et al. (2024) connect learning-rate transfer to the super consistency of landscape sharpness across model sizes. Motivated by that observation, we measure a notion of sharpness corresponding to Adam updates. This is probed during the early stage of RL post-training, after SFT on the same task.

Let the loss be $-S_\beta$ and let H_W be the Hessian with respect to a weight matrix W. Let D_W be the diagonal Adam denominator accumulated at a given step in training, held fixed during measurement. Then the measured Adam sharpness is the maximum eigenvalue of a corresponding preconditioned Hessian

$$\lambda_{\max}\left(D_W^{-1/2} H_W D_W^{-1/2}\right). \tag{4}$$

For a simple gradient step, the positive-curvature threshold would be twice the reciprocal of this eigenvalue. The plotted estimates instead use a frozen-denominator, momentum-aware AdamW local linearization, including weight decay, to obtain a positive-curvature flip boundary for each selected matrix's effective learning rate. We divide by the parameter-group learning rate multiplier in Table 1 to express that boundary in base-learning-rate units. Negative-curvature modes can preclude a strict all-mode local stability interval, so these estimates are not measured maximum stable learning rates of the full optimizer.

The results in Figure 5 show similar observations as Noci et al. (2024), and this gives us a more plausible direction to explain the observed learning rate transfer. Since the learning rate boundary is more stable under scaling, the optimizer is likely to remain more stable when using the same base learning rate across widths. As a result, we can speculate that landscape measurements are potentially more indicative of transfer properties compared to maximal update.

5 Learning Rate Transfer in RL Post-Training

We test learning rate transfer for RL on a variety of tasks, which we describe below. Links to the datasets used for these tasks are listed in Appendix F.1.

GSM8K. GSM8K evaluates multi-step arithmetic reasoning through grade-school word problems (Cobbe et al., 2021). We train on GSM8k-Aug, a synthetic augmentation with structured

arithmetic solution steps (Deng et al., 2023), and use final-answer correctness as the reinforcement learning (RL) reward.

Numina. We use NuminaMath-CoT, a mathematical reasoning corpus spanning school-level exercises and competition problems, with chain-of-thought solutions (Li et al., 2024). Supervised fine-tuning (SFT) learns these solutions, while RL rewards correctness of the extracted final answer.

Instruction following. We use prompts that combine up to five automatically verifiable output constraints, including length, keyword, and formatting requirements (Pyatkin et al., 2025). SFT uses filtered Gemma-generated responses (University of Luxembourg, 2026), and the RL reward is the fraction of constraints satisfied.

KodCode. We study Python code generation from natural-language specifications using the easy-difficulty subset of KodCode-Light-RL-10K (Xu et al., 2025). SFT uses verified DeepSeek-R1 solution code, and a binary RL reward indicates whether the generated program passes all associated unit tests.

WikiSQL. WikiSQL pairs natural-language questions with SQL queries over individual Wikipedia tables (Zhong et al., 2017). Given a question, table title when available, and column names and types, the model generates the canonical SQL query; the RL reward is exact string match after case and whitespace normalization and removal of trailing semicolons.

We study learning-rate transfer across model scale using a three-stage pipeline: pretraining, supervised fine-tuning (SFT), and reinforcement learning. We train decoder-only Transformers under either standard parameterization or CompleteP (Dey et al., 2025), retaining the chosen parameterization throughout all stages. All stages use AdamW (Loshchilov & Hutter, 2019) with linear warmup and cosine decay. For scaled parameterization, the swept learning rate η_b is the base rate before applying the prescribed parameter-specific scaling rules.

Pretraining. We initialize models from scratch and optimize the autoregressive next-token prediction objective on FineWeb-Edu (Penedo et al., 2024). We use the Llama-3.2 tokenizer and pack documents sequences of 2,048 tokens each, masking predictions across segment boundaries. As in common practice, we vary the training budget depending on the model size. Specifically, we allocate a training budget of 20 tokens per parameter. We use each model's pretrained checkpoint for task-specific SFT runs.

Supervised finetuning. For each downstream task, we finetune the pretrained model on prompt–response pairs by minimizing cross-entropy over response tokens only, masking prompt and padding tokens. All model parameters remain trainable. The task-specific SFT checkpoint initializes the corresponding reinforcement-learning sweep.

Reinforcement learning. We apply Group Relative Policy Optimization (GRPO; Shao et al., 2024) with task-specific verifiable rewards. For each prompt, we sample $G = 8$ completions at temperature 1 and compute group-normalized advantages $A_i = (r_i - \bar{r})/(\mathrm{std}(r_{1:G}) + \delta)$, where r_i is the completion reward and $\delta > 0$ ensures numerical stability. We optimize the clipped policy objective with clipping threshold 0.2 and DAPO-style token-level loss averaging (Yu et al., 2025), using one optimization iteration per rollout batch and no KL regularization.

Assessing transfer. For each task, model size, and parameterization, all GRPO learning-rate candidates start from the same task-specific SFT checkpoint and use the same data and training budget. We assess RL learning-rate transfer by comparing held-out task reward as a function of η_b across model sizes.

All training runs are done on 2 to 8 H100 GPUs, with ≈8,000 GPU hours of training. Further training details can be found in Appendix F.

Besides visually examining our experiment plots (see Figures 2, 6, and 7), we quantify the high-reward range shared across widths and containing the width-512 selected rate. In Table 2, we can see that the scaled parameterization has a wider range on all five tasks at the 80% threshold, and on three tasks at the 90% threshold. In particular, we would like to highlight the KodCode task, where the standard parameterization has no shared high-reward range at either threshold, while the scaled parameterization retains a nontrivial range at both thresholds. The KodCode sweep in Figure 6 shows that the standard parameterization has a sharp collapse in reward at larger learning rates, while the scaled parameterization retains a more stable reward across widths.

	80% of peak		90% of peak	
Task	Standard	Scaled	Standard	Scaled
GSM8K-Aug	3.02	4.14	2.71	2.48
Instruction following	3.41	3.50	2.18	1.69
KodCode	0.00	0.98	0.00	0.39
NuminaMath-CoT	1.20	1.79	0.61	1.03
WikiSQL	3.73	4.27	3.52	3.94

Table 2: Transfer range in $\log_2 \eta_b$. For the indicated threshold p and each width n, let $I(n)$ be the largest interval containing the width-512 best observed rate on which width n achieves at least p of its own best observed reward. The intersection I of $I(512)$, $I(1024)$, and $I(2048)$ has reported length $\log_2(\max I / \min I)$, or zero if empty. Interval boundaries are interpolated linearly in $\log_2 \eta_b$ within the shared swept domain.

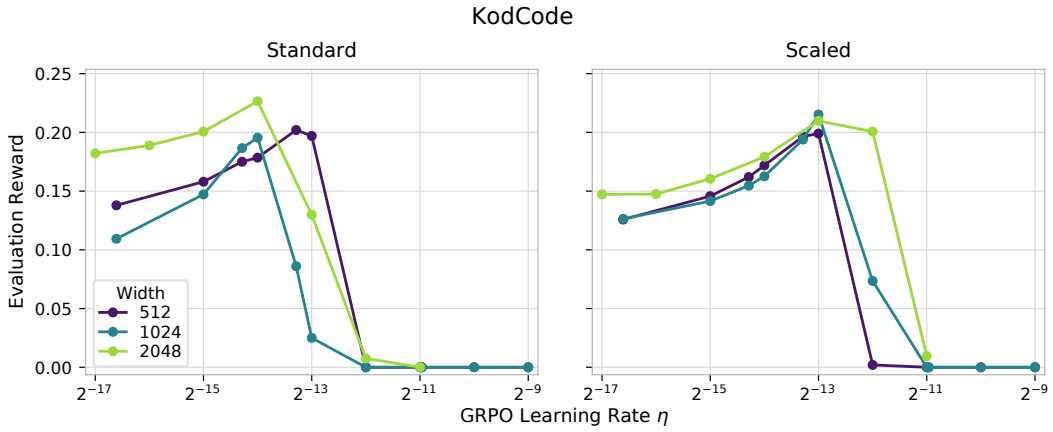


Figure 6: GRPO learning-rate sweeps on KodCode under standard parameterization (left) and scaled parameterization (right). The panels share both axes; colors denote model width.

A broad high-reward range makes learning-rate selection less sensitive to the exact sweep grid. Figure 7 shows the instruction-following comparison: at larger learning rates, the scaled wider models retain more reward than their standard counterparts. The GSM8K-Aug sweep in Figure 2 shows a sharper collapse under standard parameterization. For example, the standard width-1024 model has collapsed at $\eta_b = 2^{-13}$, whereas the width-512 model retains a reward of 0.301 at the same rate. The NuminaMath-CoT and WikiSQL sweeps are shown in Appendix E.

Table 2 measures the high-reward range shared across widths and containing the rate selected at width 512. At the 80% threshold, the scaled parameterization has a wider range on all five plotted tasks; at the 90% threshold, it has a wider range on three. The standard KodCode selection falls outside its shared high-reward range at both thresholds.

At fixed width 2048, Figure 8 compares depth transfer in GSM8K-Aug. In this case, both standard and scaled parameterizations transfer well, which echoes the observations by Dey et al. (2025) in non-depth scaled parameterizations when pretraining is done with 20 tokens per parameter.

6 Discussion

The main practical question we are addressing in this work is whether a learning rate selected on a small network remains useful when applied to a larger network, where training is significantly more expensive. The performance of this learning rate transfer approach for standard parameterization is task dependent, as some tasks show a fairly stable learning rate range across widths, while others such as GSM8K-Aug and KodCode show a weaker range (see Table 2).

With scaled parameterization, learning rates chosen for good performance on a small model remains effective for larger models on all five tasks, making tuning small models a reliable way to set learning

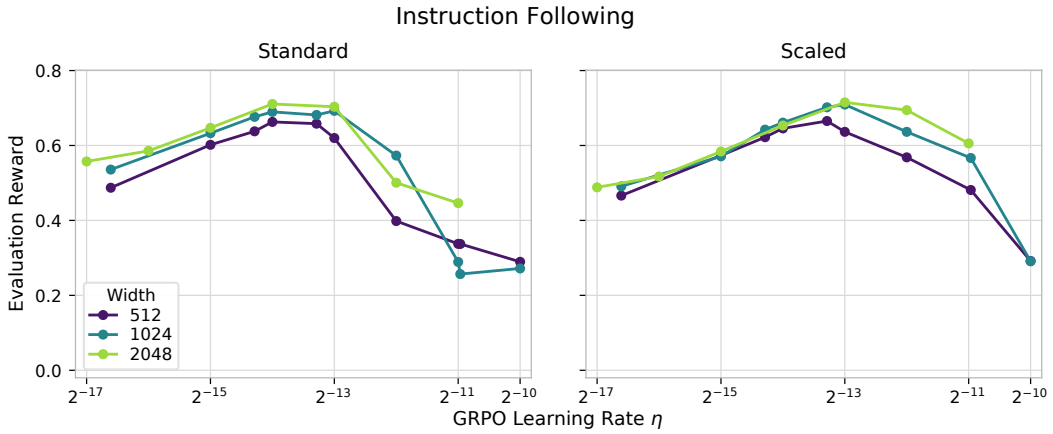


Figure 7: GRPO learning-rate sweeps for instruction following under standard parameterization (left) and scaled parameterization (right). The panels share both axes; colors denote model width.

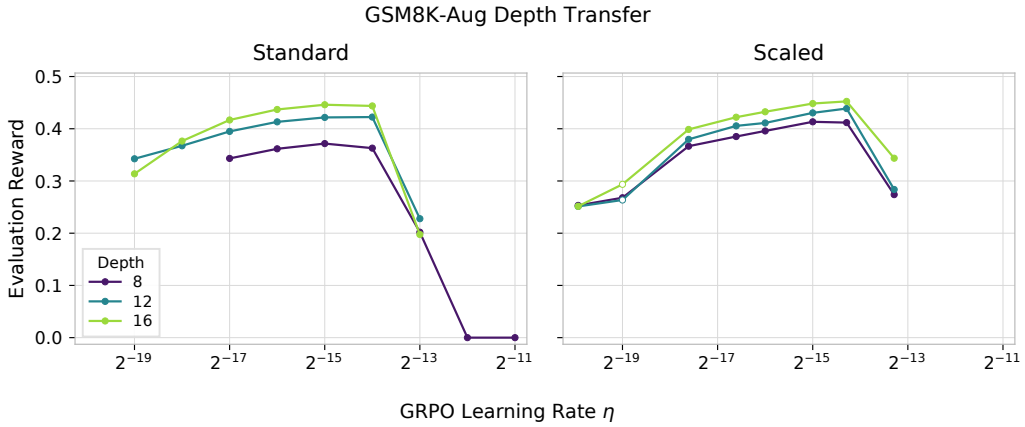


Figure 8: GRPO learning-rate sweeps on GSM8K-Aug at width 2048 for depths 8, 12, and 16 under standard parameterization (left) and scaled parameterization (right).

rates. In particular, we observe that the scaled parameterization retains a wider high-reward range across widths on all five tasks at the 80% threshold, and on three tasks at the 90% threshold (Table 2). The stability provided by the scaled parameterization is particularly reassuring for large scale post-training, where the cost is high and the performance is highly sensitive to the learning rate.

Our analysis gives a reason to expect this behavior in RL. Under the conditions of Proposition 4.1, the GRPO objective preserves the width and depth scaling. While the maximal-update measurements are not conclusive for RL (Section 3), the sharpness based boundary measurements vary far less with width under the scaled parameterization (Figure 5). This suggests that future theoretical work towards analyzing the scaling of the landscape may yield more predictive results compared to just neuron updates.

One interesting future direction is to test how far small model tuning can be pushed. Our experiments focus on a base model starting from width 512 and depth 8, but it's possible to consider even smaller models, such as width 128 and depth 2, to see if the learning rate transfer remains stable. We can also consider exploring the transfer of other hyperparameters, such as the KL regularization and clipping range.

Our results also raises many open theoretical questions, in particular how one should study multi-stage training dynamics, and especially after an extended pretraining stage. Moving away from a mean field neuron based analysis proved to be quite challenging, but may be necessary to understand the dynamics of RL post-training.

REFERENCES

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, and Sumit Sanghai. GQA: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 4895–4901, 2023. doi: 10.18653/v1/2023.emnlp-main.298. URL https://aclanthology.org/2023.emnlp-main.298/.

Johan Bjorck, Alon Benhaim, Vishrav Chaudhary, Furu Wei, and Xia Song. Scaling optimal LR across token horizons. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2409.19913.

Blake Bordelon and Cengiz Pehlevan. Self-consistent dynamical field theory of kernel evolution in wide neural networks. In *Advances in Neural Information Processing Systems*, volume 35, pp. 32240–32256, 2022. URL https://papers.nips.cc/paper_files/paper/2022/hash/d027a5c93d484a4312cc486d399c62c1-Abstract-Conference.html.

Blake Bordelon and Cengiz Pehlevan. Dynamics of finite width kernel and prediction fluctuations in mean field neural networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2024 (10):104021, 2024. URL https://arxiv.org/abs/2304.03408.

Blake Bordelon, Lorenzo Noci, Mufan Bill Li, Boris Hanin, and Cengiz Pehlevan. Depthwise hyperparameter transfer in residual networks: Dynamics and scaling limit, 2023. URL https://arxiv.org/abs/2309.16620.

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

DeepSeek-AI. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Yuntian Deng, Kiran Prasad, Roland Fernandez, Paul Smolensky, Vishrav Chaudhary, and Stuart Shieber. Implicit chain of thought reasoning via knowledge distillation, 2023. URL https://arxiv.org/abs/2311.01460.

Nolan Dey, Bin Claire Zhang, Lorenzo Noci, Mufan Li, Blake Bordelon, Shane Bergsma, Cengiz Pehlevan, Boris Hanin, and Joel Hestness. Don't be lazy: CompleteP enables compute-efficient deep transformers. In *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2505.01618.

Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks*, 107:3–11, 2018. doi: 10.1016/j.neunet.2017.12.012. URL https://arxiv.org/abs/1702.03118.

Katie E. Everett, Lechao Xiao, Mitchell Wortsman, Alexander A. Alemi, Roman Novak, Peter J. Liu, Izzeddin Gur, Jascha Sohl-Dickstein, Leslie Pack Kaelbling, Jaehoon Lee, and Jeffrey Pennington. Scaling exponents across parameterizations and optimizers. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 12666–12700, 2024. URL https://proceedings.mlr.press/v235/everett24a.html.

Cheng Ge, Caitlyn Heqi Yin, Hao Liang, and Jiawei Zhang. Why GRPO needs normalization: A local-curvature perspective on adaptive gradients, 2026. URL https://arxiv.org/abs/2601.23135.

Nikhil Ghosh, Denny Wu, and Alberto Bietti. Understanding the mechanisms of fast hyperparameter transfer, 2025. URL https://arxiv.org/abs/2512.22768.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia

Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Oriol Vinyals, Jack W. Rae, and Laurent Sifre. An empirical analysis of compute-optimal large language model training. In *Advances in Neural Information Processing Systems*, volume 35, 2022. doi: 10.52202/068431-2176. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/c1e2faff6f588870935f114ebe04a3e5-Abstract-Conference.html.

Dayal Singh Kalra and Maissam Barkeshli. Quantifying hyperparameter transfer and the importance of embedding layer learning rate, 2026. URL https://arxiv.org/abs/2605.21486.

Atli Kosson, Jeremy Welborn, Yang Liu, Martin Jaggi, and Xi Chen. Weight decay may matter more than muP for learning rate transfer in practice. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2510.19093.

Adam Lee, M. Ganesh Kumar, Blake Bordelon, and Cengiz Pehlevan. CompleteP for RL: Maintaining feature learning when scaling deep reinforcement learning. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://icml.cc/virtual/2026/poster/66419.

Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Costa Huang, Kashif Rasul, Longhui Yu, Albert Jiang, Ziju Shen, Zihan Qin, Bin Dong, Li Zhou, Yann Fleureau, Guillaume Lample, and Stanislas Polu. NuminaMath. Hugging Face dataset repository, 2024. URL https://huggingface.co/datasets/AI-MO/NuminaMath-CoT.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding R1-zero-like training: A critical perspective, 2025. URL https://arxiv.org/abs/2503.20783.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://arxiv.org/abs/1711.05101.

Meta. Llama 3.2 model card. Hugging Face model repository, 2024. URL https://huggingface.co/meta-llama/Llama-3.2-1B.

Bruno Mlodozeniec, Pierre Ablin, Louis Béthune, Dan Busbridge, Michal Klein, Jason Ramapuram, and Marco Cuturi. Completed hyperparameter transfer across modules, width, depth, batch and duration, 2025. URL https://arxiv.org/abs/2512.22382.

Lorenzo Noci, Alexandru Meterez, Thomas Hofmann, and Antonio Orvieto. Super consistency of neural network landscapes and learning rate transfer. In *Advances in Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2402.17457.

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. Training language models to follow instructions with human feedback, 2022. URL https://arxiv.org/abs/2203.02155.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben Allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro von Werra, and Thomas Wolf. The FineWeb datasets: Decanting the web for the finest text data at scale. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://arxiv.org/abs/2406.17557.

Valentina Pyatkin, Saumya Malik, Victoria Graf, Hamish Ivison, Shengyi Huang, Pradeep Dasigi, Nathan Lambert, and Hanna Hajishirzi. Generalizing verifiable instruction following. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-1645. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/46499a0622ecf568b72d17b61e45dbd5-Abstract-Datasets_and_Benchmarks_Track.html. Datasets and Benchmarks Track.

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.

Noam Shazeer. GLU variants improve transformer, 2020. URL https://arxiv.org/abs/2002.05202.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding, 2021. URL https://arxiv.org/abs/2104.09864.

University of Luxembourg. IF multi-constraints up to 5 SFT. Hugging Face dataset repository, 2026. URL https://huggingface.co/datasets/UniLu/IF_multi_constraints_upto5_SFT.

Nikhil Vyas, Alexander Atanasov, Blake Bordelon, Depen Morwani, Sabarish Sainathan, and Cengiz Pehlevan. Feature-learning networks are consistent across widths at realistic scales. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/03600ae6c3392fd65ad7c3a90c6f7ce8-Abstract-Conference.html.

Xi Wang and Laurence Aitchison. How to set AdamW's weight decay as you scale model and dataset size. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 62222–62250, 2025. URL https://proceedings.mlr.press/v267/wang25b.html.

Zhangchen Xu, Yang Liu, Yueqin Yin, Mingyuan Zhou, and Radha Poovendran. KodCode: A diverse, challenging, and verifiable synthetic dataset for coding. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 6980–7008, Vienna, Austria, July 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-acl.365. URL https://aclanthology.org/2025.findings-acl.365/.

Greg Yang and Edward J. Hu. Tensor programs IV: Feature learning in infinite-width neural networks. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 11727–11737. PMLR, 2021. URL https://proceedings.mlr.press/v139/yang21c.html.

Greg Yang, Edward J. Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. Tensor programs V: Tuning large neural networks via zero-shot hyperparameter transfer, 2022. URL https://arxiv.org/abs/2203.03466.

Greg Yang, Dingli Yu, Chen Zhu, and Soufiane Hayou. Tensor programs VI: Feature learning in infinite-depth neural networks, 2023. URL https://arxiv.org/abs/2310.02244.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. DAPO: An open-source LLM reinforcement learning system at scale, 2025. URL https://arxiv.org/abs/2503.14476.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. In *Advances in Neural Information Processing Systems*, 2019. URL https://papers.nips.cc/paper_files/paper/2019/hash/1e8a19426224ca89e83cef47f1e7f53b-Abstract.html.

Rosie Zhao, Alexandru Meterez, Sham Kakade, Cengiz Pehlevan, Samy Jelassi, and Eran Malach. Echo chamber: RL post-training amplifies behaviors learned in pretraining, 2025. URL https://arxiv.org/abs/2504.07912.

Victor Zhong, Caiming Xiong, and Richard Socher. Seq2SQL: Generating structured queries from natural language using reinforcement learning, 2017. URL https://arxiv.org/abs/1709.00103.

Hongyi Zhou, Kai Ye, Erhan Xu, Jin Zhu, Ying Yang, Shijin Gong, and Chengchun Shi. Demystifying group relative policy optimization: Its policy gradient is a U-statistic, 2026. URL https://arxiv.org/abs/2603.01162.

A AI USE DISCLOSURE

We used OpenAI Codex to help identify and assess related literature; plan, draft, and edit portions of the manuscript; and improve its presentation. We also used it to discuss experimental design, develop and debug research code, run and diagnose computational experiments, prepare figures, and analyze and interpret results. For the GRPO scaling argument, Codex assisted in examining the claim and developing and revising parts of the appendix proof. The authors take responsibility for the final text, code, results, and mathematical claims.

B RELATED WORK

Yang et al. (2022) showed that maximal-update parameterization can transfer tuned hyperparameters from small to larger networks across width. Subsequent work studied depthwise transfer and feature learning in residual networks (Bordelon et al., 2023; Yang et al., 2023). Dey et al. (2025) introduced CompleteP to combine depthwise transfer with non-lazy learning throughout deep Transformers; Mlodozeniec et al. (2025) extended this line to module-specific hyperparameters, batch size, and training duration. These studies motivate the scaling prescription used here, but primarily evaluate language-model pretraining rather than task-specific supervised fine-tuning or policy optimization.

Several optimizer choices also affect observed transfer. Everett et al. (2024) showed that several parameterizations can achieve transfer when layerwise learning rates and optimizer conventions are chosen appropriately. Kalra & Barkeshli (2026) identified the embedding learning rate as a major factor in AdamW pretraining transfer, while Kosson et al. (2026) found that weight decay and warmup can stabilize width-dependent update dynamics. Related analyses connect transfer to training horizon, landscape sharpness, and the quality of a transferred rate rather than only the location of the optimum (Bjorck et al., 2025; Noci et al., 2024; Ghosh et al., 2025). These results motivate evaluating both transfer performance and update or stability diagnostics across training stages.

GRPO, introduced by Shao et al. (2024), uses group-relative advantages with a clipped policy objective rooted in PPO (Schulman et al., 2017). Yu et al. (2025) proposed DAPO, including token-level loss aggregation; our GRPO experiments use this aggregation without adopting its full training recipe. Analyses of response-length bias, reward normalization, and finite-group effects show that implementation and rollout statistics can alter optimization behavior (Liu et al., 2025; Ge et al., 2026; Zhou et al., 2026). Consequently, a model-size-neutral algebraic form of the objective alone does not guarantee that its observed best learning rate will transfer.

Most directly, Lee et al. (2026) studied CompleteP transfer and feature learning in continuous-control reinforcement learning. Our study follows autoregressive language models through pretraining, task-specific supervised fine-tuning, and verifiable-reward GRPO, with a separately tuned base learning rate within each stage. Multistage language-model pipelines and controlled studies of how pretraining influences later reinforcement learning provide context for that design (Ouyang et al., 2022; DeepSeek-AI, 2025; Zhao et al., 2025). We examine whether a rate selected at smaller scale remains effective as width or depth changes, and how stage transitions affect the diagnostics underlying that transfer.

C PROOF OF THE GRPO SCALING PROPOSITION

Let θ denote the current model parameters, q a prompt, and o a completion with tokens $o_1, \ldots, o_{|o|}$. Write $f_\theta(q, o_{<t})_v$ for the v-th output logit at the last position of the prefix $(q, o_{<t})$. The autoregressive policy is

$$\pi_\theta(o \mid q) = \prod_{t=1}^{|o|} \pi_\theta(o_t \mid q, o_{<t}), \qquad \pi_\theta(v \mid q, o_{<t}) = \frac{\exp(f_\theta(q, o_{<t})_v)}{\sum_{u=1}^{n_0} \exp(f_\theta(q, o_{<t})_u)}, \quad v \in [n_0]. \tag{5}$$

Let μ denote the autoregressive rollout policy, fixed during an update. The token importance ratio is

$$r_\theta(o_t \mid q, o_{<t}) := \frac{\pi_\theta(o_t \mid q, o_{<t})}{\mu(o_t \mid q, o_{<t})}. \tag{6}$$

For a prompt q, sample $G \geq 2$ completions $o_1, \ldots, o_G$ from μ and let $r_i = R(q, o_i)$ be their rewards. Write $|o_i|$ for the number of valid completion tokens, excluding padding, and let $M = \sum_{i=1}^{G} |o_i| > 0$. Their empirical mean is $\bar{r} = G^{-1} \sum_{i=1}^{G} r_i$, and $\mathrm{std}(r_{1:G})$ is their empirical standard deviation. The group advantage is $A_i = (r_i - \bar{r})/(\mathrm{std}(r_{1:G}) + \delta)$, with $\delta > 0$. Abbreviate the token ratio by $r_{\theta,i,t} := r_\theta(o_{i,t} \mid q, o_{i,<t})$. With clipping threshold ε (set to 0.2 in our runs), the empirical objective for one prompt group is

$$S_0(\theta; \mu \mid q) = \frac{1}{M} \sum_{i=1}^{G} \sum_{t=1}^{|o_i|} \min\big\{ r_{\theta,i,t} A_i, \mathrm{clip}\big(r_{\theta,i,t}, 1-\varepsilon, 1+\varepsilon\big) A_i \big\}. \tag{7}$$

Let π_{ref} be a fixed reference policy with positive next-token probabilities. Define the next-token KL divergence at each sampled prefix by

$$D_{i,t} := \sum_{v=1}^{n_0} \pi_\theta(v \mid q, o_{i,<t}) \log \frac{\pi_\theta(v \mid q, o_{i,<t})}{\pi_{\mathrm{ref}}(v \mid q, o_{i,<t})}. \tag{8}$$

For fixed $\beta \geq 0$, define

$$S_\beta(\theta; \mu \mid q) := S_0(\theta; \mu \mid q) - \beta K(\theta; \mu \mid q), \qquad K(\theta; \mu \mid q) := \frac{1}{M} \sum_{i=1}^{G} \sum_{t=1}^{|o_i|} D_{i,t}. \tag{9}$$

Unlike the sequence-level KL in the idealized objective, K averages next-token KL divergences over sampled rollout prefixes. In a minibatch, DAPO-style aggregation sums the numerator and valid-token denominator over prompt groups (Yu et al., 2025). Our experiments set $\beta = 0$.

We differentiate with the sampled group and rollout policy fixed, viewing the logits at each token occurrence as separate outputs. Let Ω be the token indices where clipping makes the reward term constant:

$$\begin{aligned} \Omega = \{(i,t) : A_i > 0,\ r_{\theta,i,t} \geq 1+\varepsilon\} \\ \cup \{(i,t) : A_i < 0,\ r_{\theta,i,t} \leq 1-\varepsilon\}. \end{aligned} \tag{10}$$

The softmax derivative, with μ fixed, is

$$\begin{aligned} \frac{\partial \log \pi_\theta(o_{i,t} \mid q, o_{i,<t})}{\partial f_\theta(q, o_{i,<t})_v} &= \mathbf{1}_{\{v = o_{i,t}\}} - \pi_\theta(v \mid q, o_{i,<t}), \\ \frac{\partial r_{\theta,i,t}}{\partial f_\theta(q, o_{i,<t})_v} &= r_{\theta,i,t} \left(\mathbf{1}_{\{v = o_{i,t}\}} - \pi_\theta(v \mid q, o_{i,<t}) \right). \end{aligned} \tag{11}$$

Away from the clipping thresholds, this gives

$$\frac{\partial S_0(\theta; \mu \mid q)}{\partial f_\theta(q, o_{i,<t})_v} = \frac{\mathbf{1}_{\{(i,t) \notin \Omega\}} A_i r_{\theta,i,t}}{M} \left(\mathbf{1}_{\{v = o_{i,t}\}} - \pi_\theta(v \mid q, o_{i,<t}) \right). \tag{12}$$

Summing the absolute values over $v \in [n_0]$ gives

$$\|\nabla_f S_0\|_1 = \frac{2}{M} \sum_{i=1}^{G} \sum_{\substack{1 \leq t \leq |o_i| \\ (i,t) \notin \Omega}} |A_i| r_{\theta,i,t} \big(1 - \pi_\theta(o_{i,t} \mid q, o_{i,<t})\big). \tag{13}$$

At a clipping threshold, the upper bound below holds for any subgradient.

Similarly, differentiating the KL term gives

$$\frac{\partial D_{i,t}}{\partial f_\theta(q, o_{i,<t})_v} = \pi_\theta(v \mid q, o_{i,<t}) \left(\log \frac{\pi_\theta(v \mid q, o_{i,<t})}{\pi_{\mathrm{ref}}(v \mid q, o_{i,<t})} - D_{i,t} \right). \tag{14}$$

The scaled parameterization keeps within-prefix logit ranges bounded as n, d vary, as described in Section 2. We use the same bound for the rollout policy and, when $\beta > 0$, the reference policy at the evaluated prefixes. Since G and $n_0 \geq 2$ are fixed, their next-token probabilities are bounded below by a constant $c > 0$. Hence $c \leq r_{\theta,i,t} \leq c^{-1}$. When the KL term is present, the log

ratios between current and reference probabilities are bounded in absolute value by $\log(1/c)$. The empirical normalization also gives $|A_i| \le \sqrt{G}$. Equations (13) and (14) now yield

$$\|\nabla_f S_\beta\|_1 \le \frac{2\sqrt{G}}{c} + 2\beta \log(1/c) = O(1) \quad \text{as } n, d \text{ vary.} \tag{15}$$

The normalization by M removes an extra factor of response length; the same calculation applies to DAPO minibatch aggregation.

Conversely, suppose the token average $M^{-1} \sum_{(i,t)\notin\Omega} |A_i|$ stays bounded away from zero. The probability bound above gives $1 - \pi_\theta(o_{i,t} \mid q, o_{i,<t}) \ge (n_0 - 1)c$, so (13) yields $\|\nabla_f S_0\|_1 = \Omega(1)$. For $\beta > 0$ the conclusion persists when the KL gradient does not asymptotically cancel the reward gradient. Together with the upper bound, this gives the stated scaling in the nondegenerate regime.

D Additional Early-RL Base Learning-Rate Boundaries

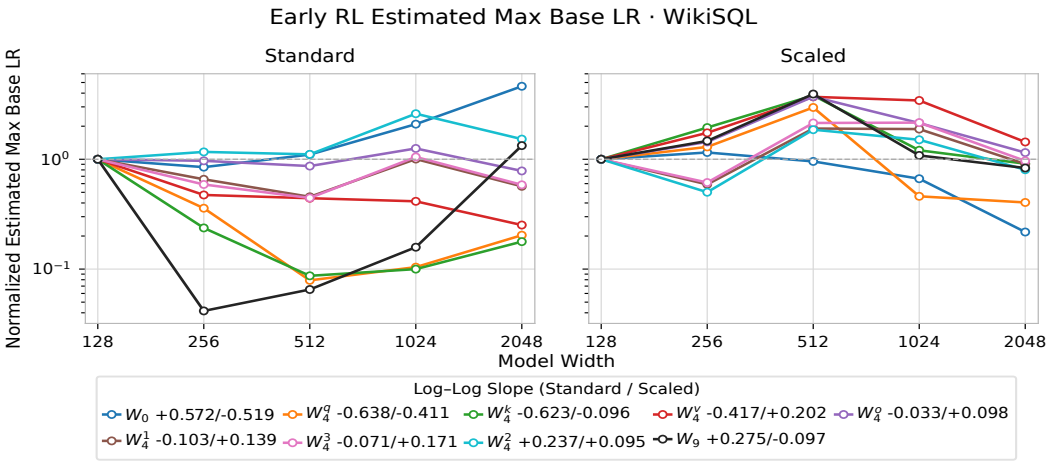


Figure 9: Estimated base learning-rate boundary after five GRPO warmup updates on WikiSQL. Each selected-matrix positive-curvature flip boundary is normalized to its width-128 value within each parameterization, as in Figure 5. Several scaled curves vary appreciably with width. Open markers indicate that the boundary is not a strict all-mode local stability interval.

E Additional GRPO Learning-Rate Sweeps

The remaining task sweeps use the same axes and width colors for the standard and scaled parameterizations as the figures in the main text.

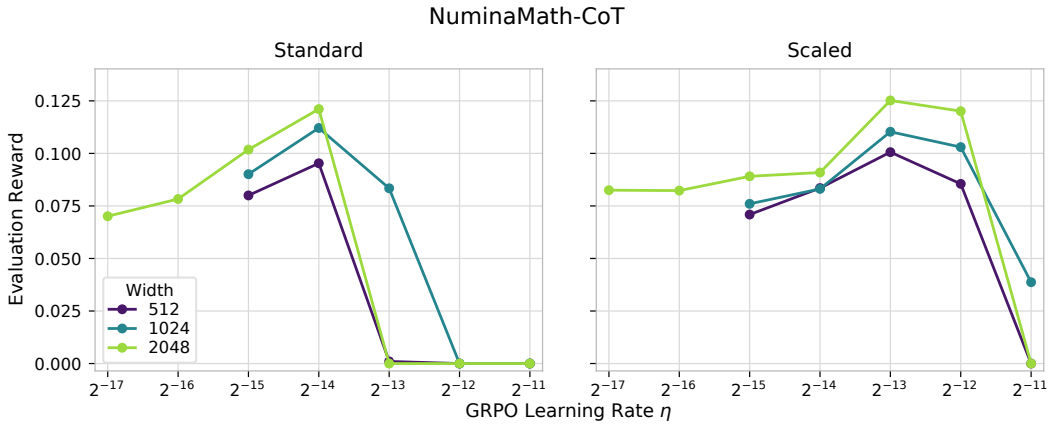


Figure 10: GRPO learning-rate sweeps on NuminaMath-CoT. Standard parameterization is shown on the left and the scaled parameterization on the right; the panels share both axes.



Figure 11: GRPO learning-rate sweeps on WikiSQL. Standard parameterization is shown on the left and the scaled parameterization on the right; the panels share both axes.

F Further Training Details

F.1 Dataset Links

The datasets described in Section 5 are available at:

- GSM8K-Aug: `https://huggingface.co/datasets/zen-E/GSM8k-Aug`
- Instruction-following RL prompts: `https://huggingface.co/datasets/allenai/IF_multi_constraints_upto5`
- Instruction-following SFT responses: `https://huggingface.co/datasets/UniLu/IF_multi_constraints_upto5_SFT`
- KodCode-Light-RL-10K: `https://huggingface.co/datasets/KodCode/KodCode-Light-RL-10K`

Table 3: Training hyperparameters for the three stages. Unless noted, settings are shared by the standard-parameterization (SP) and CompleteP (CP) arms. For CP, η_0 and λ_0 denote the *base* learning rate and weight decay before CompleteP's parameter-specific scaling. Task-dependent values are listed in Table 5; per-model pretraining budgets in Table 4.

	Pretraining	**SFT**	**GRPO**
Data and objective			
Data	FineWeb-Edu (`sample-10BT`)	prompt–response pairs	prompts + verifiable reward
Objective	next-token CE	CE on response tokens	clipped policy gradient (DAPO)
Initialization	random	pretrained checkpoint	selected SFT checkpoint
Sequence length	2,048 (packed)	512–1,536	prompt 256–1,024, compl. 128–1,024
Optimization			
Optimizer	AdamW (β_1=0.9, β_2=0.999, ϵ=10^{-8})		
LR schedule	linear warmup, cosine decay to a fixed fraction of the peak		
Warmup steps	500	100–200	100
Final LR / peak LR	0.1	0.1	0.5
Peak LR η_0, CP	10^{-3} (d≤1024); 3×10^{-3} (d=2048)	per task, shared across d	swept per task and d
Peak LR η_0, SP	5×10^{-4}	per task and d	swept per task and d
LR grid	$\{5, 10, 30\}\times10^{-4}$ (CP only)	$\{1, 3, 5, 10, 30\}\times10^{-4}$	2^{-k}, $k \in \{9, \ldots, 19\}$
Weight decay	SP 0.1; CP λ_0 (Table 4)	0.01	0
Grad. clip (global norm)	1.0	none	1.0
Global batch	256 seq. (0.52M tokens)	128 or 256 seq.	128 completions (16 prompts $\times$ G)
Training steps	9.7k–57.2k (Table 4)	400–5,000	500–5,000
Precision	bf16 mixed precision, activation checkpointing		
Hardware	8× GPU	2× GPU	2× GPU
Random seed	42		
GRPO-specific			
Group size G	–	–	8
Sampling	–	–	T=1.0, top-p=1.0, no top-k
Clip threshold ε	–	–	0.2
KL coefficient β	–	–	0 (no reference model)
Advantage	–	–	group-normalized (mean/std over G)
Importance ratio	–	–	token-level
Updates per rollout batch	–	–	1 (on-policy)

Table 4: Pretraining budgets and CompleteP base weight decay for each model size. All models share the Llama-3 block (RMSNorm, SwiGLU, RoPE θ=5×10^5), 32 query heads with 8 KV heads of dimension 64, MLP width 8,192, the Llama-3.2 tokenizer (vocabulary 128,256), and untied input/output embeddings; width d and depth L are varied. Tokens = steps $\times$ 256 $\times$ 2,048. CompleteP scales the hidden-matrix decay as $\lambda_0\, d/d_0$ with d_0=512, so the effective decay depends only on depth (0.114, 0.092, 0.077 for L=8, 12, 16); SP uses λ=0.1 throughout.

d	L	Steps	Tokens	CP λ_0
512	8	9,650	5.1B	0.114
512	12	11,970	6.3B	0.092
512	16	14,291	7.5B	0.077
1024	8	19,301	10.1B	0.057
1024	12	23,941	12.6B	0.046
1024	16	28,581	15.0B	0.038
2048	8	38,601	20.2B	0.028
2048	12	47,882	25.1B	0.023
2048	16	57,163	30.0B	0.019

Table 5: Task-specific post-training settings and learning-rate sweeps (depth L=8, widths $d \in \{512, 1024, 2048\}$). NuminaMath is NuminaMath-CoT, IF is multi-constraint instruction following (up to five constraints), KodCode is the easy split of KodCode-Light-RL-10K. SFT learning rates were swept at d=512 and 1024; under CompleteP the selected base rate is reused unchanged at d=2048, whereas the SP arm uses 5×10^{-4} at d=2048 without a sweep. GRPO sweeps every power of two in the listed interval at each width; at $d\leq1024$ a coarser decade grid (10^{-6}–10^{-3}) was run first. Rewards are binary except IF, which uses the fraction of constraints satisfied. Other GRPO settings are fixed as in Table 3.

	GSM8K-Aug	**NuminaMath**	**WikiSQL**	**IF**	**KodCode**
Supervised finetuning (learning rates in units of 10^{-4})					
Steps	3,000	5,000	1,000	1,000	400
Global batch (sequences)	256	128	256	128	128
Max sequence length	512	1,024	512	1,536	1,024
Warmup steps	200	200	200	100	100
LR grid	$\{5, 10, 30\}$	$\{5, 10, 30\}$	$\{1, 5, 10\}$	$\{1, 3, 5, 10\}$	$\{1, 3, 10\}$
Selected LR, CP (all d)	30	30	5	5	10
Selected LR, SP (d=512/1024/2048)	10 / 5 / 5	10 / 10 / 5	5 / 5 / 5	5 / 5 / 5	10 / 3 / 5
GRPO					
Steps	3,000 (SP: 5,000)	500	3,000	1,000	1,000
Max prompt / completion tokens	512 / 256	512 / 1,024	256 / 128	512 / 1,024	1,024 / 512
LR grid, $d \in \{512, 1024\}$	$[2^{-15}, 2^{-9}]$	$[2^{-15}, 2^{-11}]$	$[2^{-15}, 2^{-11}]$	$[2^{-15}, 2^{-10}]$	$[2^{-15}, 2^{-9}]$
LR grid, d=2048			$[2^{-17}, 2^{-11}]$		
Reward	final answer	boxed answer	SQL exact match	constraints satisfied	unit tests pass