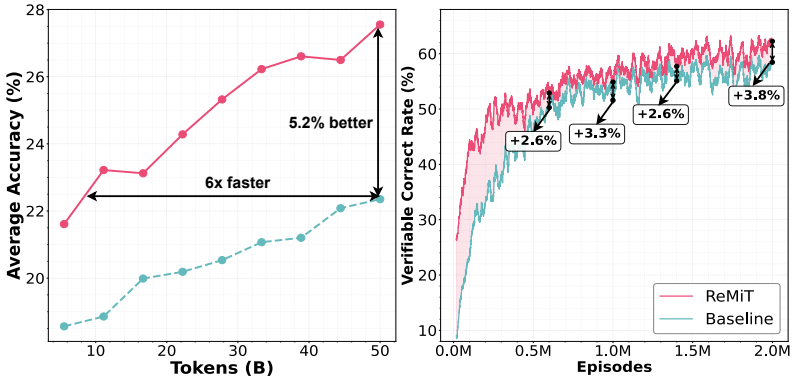
# RL-GUIDED MID-TRAINING FOR ITERATIVE LLM EVOLUTION

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Standard training pipelines for large language models (LLMs) are typically unidirectional, progressing from pre-training to post-training. However, the potential for a bidirectional process—where insights from post-training retroactively improve the pre-trained foundation—remains unexplored. We aim to establish a **self-reinforcing flywheel**: a cycle in which reinforcement learning (RL)-tuned model strengthens the base model, which in turn enhances subsequent post-training performance, requiring no specially trained teacher model. To realize this, we analyze training dynamics and identify the mid-training phase as a critical turning point for model capabilities. This phase typically occurs at the end of pre-training, utilizing high-quality corpora under a rapidly decaying learning rate. Building upon this insight, we introduce **ReMiT** (**Re**inforcement Learning-Guided **Mi**d-**T**raining) as a concrete feedback mechanism, which leverages the reasoning priors of RL models to dynamically reweight tokens during the mid-training phase. Empirically, ReMiT achieves an average improvement of 3% on 10 pre-training benchmarks, spanning math, code, and general reasoning, and sustains these gains by over 2% throughout the post-training pipeline. These results validate an iterative feedback loop, enabling self-reinforcing evolution of LLMs.




(a) Performance Comparison at the Mid-Training Stage on OLMo-1B

(b) Performance Comparison at the RL Stage on OLMo-1B

Figure 1: (a) Average score on the ten-task mid-training suite as a function of training tokens. (b) Verifiable correct rate during subsequent RL.

## 1 INTRODUCTION

Training a language model typically proceeds through multiple stages whose objectives provide distinct forms of supervision (Achiam et al., 2023; Touvron et al., 2023; Yang et al., 2025a). Pre-training exposes the model to diverse contexts and prediction targets, whereas post-training leverages demonstrations, preference data, or reward signals to shape its task behavior (Schulman et al., 2017; Shao et al., 2024). These stages are conventionally organized as a sequential pipeline, with the model produced by each stage serving as the initialization for the next.



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.

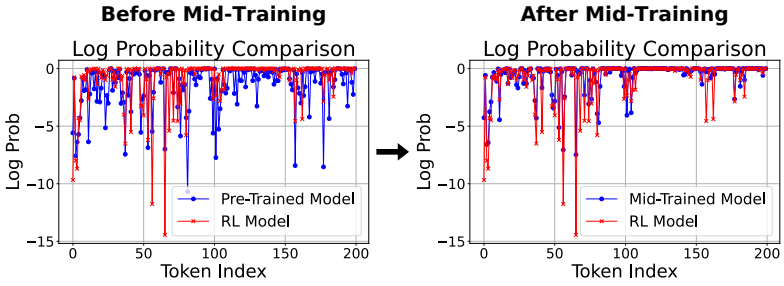



Figure 2: We identify the mid-training stage as a critical turning point, as it rapidly shifts the base model's token distribution toward that of a more capable RL model.

Prior work on reference-guided pre-training has shown that model-derived signals can improve corpus learning. For example, RHO-1 uses excess loss computed with a reference model to select training tokens (Lin et al., 2024), while MiniPLM uses teacher signals to refine the pretraining data distribution (Gu et al., 2024). More recently, Tan et al. (2026) use post-trained models to rewrite corpus continuations and judge generated sequences. Assessing the value of such guidance also requires considering how its effects carry over to subsequent post-training. Recent work, including OctoThinker (Wang et al., 2025c), shows that training choices made toward the end of pre-training can substantially affect subsequent reinforcement learning (RL), motivating evaluation beyond immediate base-model performance.

Building on these observations, we consider the post-trained model itself as a source of guidance for pre-training. Having been shaped by task-oriented supervision, this model may provide learning signals relevant to the downstream objectives of the training pipeline, which motivates our central research question:

**Can feedback from a post-trained model guide pre-training and improve the resulting model's performance after subsequent post-training?**

We investigate this question through mid-training, the late phase of pre-training that refines a model on high-quality corpora under a rapidly decaying learning rate (Touvron et al., 2023; Yang et al., 2025a); Appendix A provides further background. Our analysis shows that the model's token-level predictive distribution becomes closer to that of the final RL model during this stage, as illustrated in Figure 2 and further examined in Section 2.2. These observations motivate mid-training as a practical point for introducing guidance from an RL-tuned model.

A key design choice is how to translate the RL model's information into guidance for corpus learning. Knowledge distillation (KD) provides one mechanism through distribution matching. In this work, we introduce an alternative feedback mechanism, ReMiT (**Re**inforcement Learning-Guided **Mi**d-**T**raining), which converts relative reference–student prediction discrepancies into bounded token weights. These weights allow the RL reference to influence the relative emphasis of corpus learning while retaining the observed next-token prediction targets. ReMiT thus incorporates signals shaped by task-oriented post-training into the existing pre-training pipeline, aiming to build a stronger foundation for subsequent post-training and improve the final model's performance.

The value of this guidance is assessed through the model obtained after subsequent post-training. We therefore compare ReMiT and KD as alternative feedback mechanisms and examine how their effects evolve across training stages. We further explore feedback reuse on OLMo (OLMo et al., 2024) by using the resulting RL model to guide a second round of mid-training. This forms a training flywheel in which a post-trained model serves both as the output of one training cycle and as the reference for the next.

In summary, the contributions of this paper are as follows:

- **Feedback from post-training to corpus learning:** We introduce a training framework that reuses RL-tuned models to guide mid-training, feeding task-informed signals from post-training back into corpus learning.

- **A concrete feedback mechanism:** We develop ReMiT, which converts relative reference–student prediction discrepancies into bounded token weights while retaining the next-token prediction targets.
- **Empirical validation:** We demonstrate gains after subsequent post-training across multiple model families and characterize how different feedback mechanisms affect performance across training stages.

## 2 Preliminaries and Empirical Motivation

### 2.1 Standard Mid-Training Formulation

We consider mid-training with the next-token prediction (NTP) objective on a corpus $\mathcal{D}_m$. For a sequence $x = (x_1, \ldots, x_n)$ from the mid-training corpus $\mathcal{D}_m$, let $\ell_{\theta,t} = -\log p_\theta(x_t \mid x_{<t})$ denote the token-level loss. The objective is

$$\mathcal{L}_{\text{NTP}}(\theta) = \mathbb{E}_{x\sim\mathcal{D}_m}\left[\frac{1}{n}\sum_{t=1}^{n}\ell_{\theta,t}\right]. \tag{1}$$

Throughout, the expectation uses length-proportional sequence weights, yielding a uniform average over valid corpus tokens.

### 2.2 Motivation for Feedback During Mid-Training

Mid-training sits at the transition from broad corpus learning to task-oriented post-training. Feedback introduced at this point can act on an established knowledge foundation while shaping the initialization for subsequent optimization. Prior work shows that choices made during mid-training can influence later RL (Wang et al., 2025c).

To examine how predictions evolve during this stage, Figure 2 compares the log probabilities assigned to the same observed tokens and prefixes by checkpoints before and after mid-training and by an RL reference. For the sequence shown, the mid-trained checkpoint more closely tracks the reference, while discrepancies remain uneven across token positions. This variation motivates considering token-level prediction differences as a source of guidance for corpus learning. Appendix C provides additional visualizations of the prediction differences between the base and RL models.

## 3 RL Feedback for Mid-Training

We use an RL model to guide mid-training and assess this guidance through the model obtained after subsequent post-training. We first formalize this feedback connection, then introduce ReMiT and analyze how its token weights modify corpus learning.

### 3.1 Feedback Across Training Stages

Let $\theta_0$ denote the checkpoint before mid-training, $\mathcal{D}_m$ the mid-training corpus, and $T$ a frozen RL-tuned reference model. A guidance mechanism $G$ specifies how information from $T$ enters the student's mid-training objective. The resulting training process is

$$\theta_{\text{mid}}^{T,G} = \mathcal{M}_G(\theta_0; \mathcal{D}_m, T), \tag{2}$$

$$\theta_{\text{final}}^{T,G} = \mathcal{P}_K(\theta_{\text{mid}}^{T,G}; \mathcal{D}_p), \tag{3}$$

where $\mathcal{M}_G$ denotes guided mid-training and $\mathcal{P}_K$ denotes the subsequent post-training procedure with budget $K$ and training data and task distributions $\mathcal{D}_p$. Within each model family, comparisons use the same $\theta_0$, mid-training corpus and token budget, and subsequent post-training recipe.

We assess guidance through the value of the resulting initialization for subsequent training. Given a downstream evaluation metric $J$, where larger values indicate better performance, define

$$V_K(\theta) = J\big(\mathcal{P}_K(\theta; \mathcal{D}_p)\big). \tag{4}$$

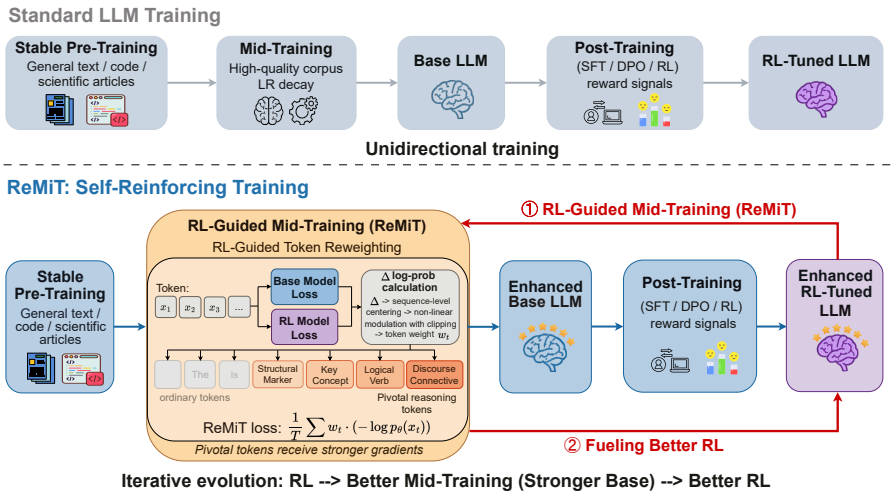



Figure 3: Overview of the proposed ReMiT framework. The pipeline connects pre-training and post-training, establishing a self-reinforcing flywheel.

This value captures performance after the specified post-training procedure. The benefit of guidance relative to standard NTP mid-training is then

$$\Delta_K(T, G) = V_K\left(\theta_{\text{mid}}^{T,G}\right) - V_K\left(\theta_{\text{mid}}^{\text{NTP}}\right), \tag{5}$$

where $\theta_{\text{mid}}^{\text{NTP}}$ is obtained under the same mid-training setup using the standard NTP objective. Positive $\Delta_K(T, G)$ indicates an improvement in the final training outcome under the evaluated recipe.

This formulation separates the reference from the transfer mechanism: KD matches reference predictions, whereas ReMiT reweights observed corpus-token losses. The initial reference is an available RL-tuned checkpoint from the corresponding model family. In the feedback-reuse experiment, the RL model produced by one training cycle serves as the reference for the next; each mid-training run restarts from the same $\theta_0$.

### 3.2 ReMiT: Bounded Token Reweighting

ReMiT uses reference predictions to adjust token weights while retaining the observed corpus targets. Using the token losses defined in Section 2.1, let $\ell_{T,t} = -\log p_T(x_t \mid x_{<t})$ denote the reference loss. We measure the discrepancy by

$$\delta_t = \ell_{\theta,t} - \ell_{T,t} = \log \frac{p_T(x_t \mid x_{<t})}{p_\theta(x_t \mid x_{<t})}. \tag{6}$$

Both models condition on the same observed prefix. A positive $\delta_t$ means that the reference assigns higher probability to the observed token.

**From relative discrepancies to bounded weights.** We remove the sequence-wide discrepancy offset by centering:

$$\widehat{\delta}_t = \delta_t - \bar{\delta}, \qquad \bar{\delta} = \frac{1}{n}\sum_{j=1}^{n} \delta_j. \tag{7}$$

The token weights are

$$w_t = \text{clip}\left(2\sigma(\widehat{\delta}_t), 1-\epsilon, 1+\epsilon\right), \tag{8}$$

where $\sigma$ is the sigmoid function and $0 < \epsilon < 1$. Since $2\sigma(0) = 1$, discrepancies above the sequence mean receive weights greater than one, and those below it receive weights

less than one. The positive lower bound retains supervision at every token, while clipping limits its rescaling. We use $\epsilon = 0.2$. Centering does not enforce mean-one weights after the nonlinear mapping, so the weights can affect both relative token emphasis and the overall scale of a sequence's loss.

**Training objective and gradient correction.** The RL reference remains frozen. We recompute weights as the student evolves and detach them during backpropagation. Using the same averaging convention as Equation 1, the objective is

$$\mathcal{L}_{\text{ReMiT}}(\theta; T) = \mathbb{E}_{x\sim\mathcal{D}_m}\left[\frac{1}{n}\sum_{t=1}^{n} \text{sg}(w_t)\,\ell_{\theta,t}\right], \tag{9}$$

where sg denotes stop-gradient. Let $g_t = \nabla_\theta \ell_{\theta,t}$. The gradient used for each update decomposes as

$$\begin{aligned}\nabla_\theta \mathcal{L}_{\text{ReMiT}} &= \mathbb{E}_{x\sim\mathcal{D}_m}\left[\frac{1}{n}\sum_{t=1}^{n} w_t g_t\right] \\ &= \nabla_\theta \mathcal{L}_{\text{NTP}} + \mathbb{E}_{x\sim\mathcal{D}_m}\left[\frac{1}{n}\sum_{t=1}^{n}(w_t - 1) g_t\right].\end{aligned} \tag{10}$$

Thus, ReMiT adds a reference-dependent correction to the NTP gradient. Since $|w_t - 1| \le \epsilon$, the triangle inequality gives

$$\|\nabla_\theta \mathcal{L}_{\text{ReMiT}} - \nabla_\theta \mathcal{L}_{\text{NTP}}\| \le \epsilon\, \mathbb{E}_{x\sim\mathcal{D}_m}\left[\frac{1}{n}\sum_{t=1}^{n}\|g_t\|\right]. \tag{11}$$

This bound controls the correction in terms of individual token-gradient norms. We evaluate its downstream benefit through $\Delta_K(T, \text{ReMiT})$ under the specified post-training procedure.

### 3.3 Local Analysis of Feedback Utility

Equation 11 controls the magnitude of ReMiT's gradient correction. Its downstream benefit also depends on the direction of that correction. We examine this dependence through a local corpus update followed by the fixed post-training procedure.

**Local utility of a token update.** Let $z = (x, t)$ denote a position in an observed sequence, with $\mathbb{E}_z$ following the corpus-token averaging convention in Section 2.1. Write $g(z) = \nabla_\theta \ell_\theta(z)$ and let $w(z)$ be the detached weight evaluated at the current parameters. For this local analysis, let $\widetilde{V}_K$ be a differentiable surrogate for the post-training value in Equation 4. Assume its gradient is $L$-Lipschitz in a convex neighborhood containing the updates considered.

Define the local utility of a token update as

$$a_K(z) = -\left\langle \nabla_\theta \widetilde{V}_K(\theta), g(z)\right\rangle. \tag{12}$$

A positive $a_K(z)$ means that a sufficiently small gradient step on this token increases the surrogate value after subsequent post-training. This quantity characterizes an update's downstream effect; it is not computed by ReMiT.

At the same parameter point, the uniform and weighted gradient directions are

$$u_0 = -\mathbb{E}_z[g(z)], \qquad u_w = -\mathbb{E}_z[w(z)g(z)]. \tag{13}$$

We compare the initializations $\theta + \eta u_0$ and $\theta + \eta u_w$ for a step size $\eta > 0$.

**Separating token emphasis from update scale.** A first-order expansion with a bounded remainder gives

$$\begin{aligned}\widetilde{V}_K(\theta+\eta u_w)-\widetilde{V}_K(\theta+\eta u_0)\\ =\eta\,\mathbb{E}_z[(w-1)a_K]+R_\eta\\ =\eta\left[\mathrm{Cov}_z(w,a_K)+\big(\mathbb{E}_z[w]-1\big)\mathbb{E}_z[a_K]\right]+R_\eta,\end{aligned}\tag{14}$$

where

$$|R_\eta|\le\frac{L\eta^2}{2}\left(\|u_w\|^2+\|u_0\|^2\right).\tag{15}$$

The derivation is provided in Appendix B.

Equation 14 distinguishes two effects. The covariance term measures whether larger weights favor token updates with greater downstream utility. The second term captures the contribution of changing the mean weight relative to uniform NTP. Both terms matter for ReMiT because sequence centering does not enforce mean-one weights. If their sum is positive, weighted training has a local advantage for sufficiently small step sizes. More precisely, a sufficient condition is

$$\begin{aligned}\mathrm{Cov}_z(w,a_K)+\big(\mathbb{E}_z[w]-1\big)\mathbb{E}_z[a_K]\\ >\frac{L\eta}{2}\left(\|u_w\|^2+\|u_0\|^2\right).\end{aligned}\tag{16}$$

**Implications for reference guidance.** ReMiT uses the centered reference–student discrepancy as an observable signal for allocating training emphasis. The analysis identifies the correspondence this signal should achieve: higher weights should favor updates with greater post-training utility, accounting for the accompanying change in scale. Clipping bounds the rescaling but does not establish this correspondence. We therefore assess the signal through the final-stage comparisons defined by $\Delta_K(T,G)$.

This result characterizes a single Euclidean gradient step at a shared parameter point. It provides a local interpretation of feedback utility; the effects of repeated updates and the actual optimizer are evaluated through the complete training pipeline.

## 4 Experiments

We evaluate whether RL-derived guidance improves the model obtained after subsequent post-training, examine its effects across training stages, and analyze the reference and weighting choices underlying ReMiT.

### 4.1 Experimental Setup

**Models and training coverage.** We evaluate mid-training on three model families: OLMo-1B (OLMo et al., 2024), SmolLM3-3B (SmolLM et al., 2025), and Youtu-LLM-2B (Lu et al., 2025). Each run starts from the corresponding checkpoint before mid-training and uses the family's publicly released RL-tuned model as the initial reference. Subsequent post-training is evaluated on two families: OLMo-1B undergoes SFT, DPO, and RL with verifiable rewards, while Youtu-LLM-2B undergoes SFT and RL. We additionally evaluate direct RL after mid-training and reuse of an updated reference on OLMo-1B.

**Baselines.** We compare ReMiT with uniform NTP, MiniPLM-style offline sample selection (Gu et al., 2024), and RHO-1-style token selection (Lin et al., 2024). The selection baselines use a retention ratio of 0.8. Our RHO-1 implementation uses a reference trained on curated data, whereas ReMiT uses an RL-tuned reference. On OLMo-1B and Youtu-LLM-2B, we also evaluate KD using the RL reference and SFT-guided token weighting. KD provides an alternative mechanism for transferring reference information; SFT-guided weighting examines the choice of reference. We report the checkpoint before mid-training as an additional comparison.

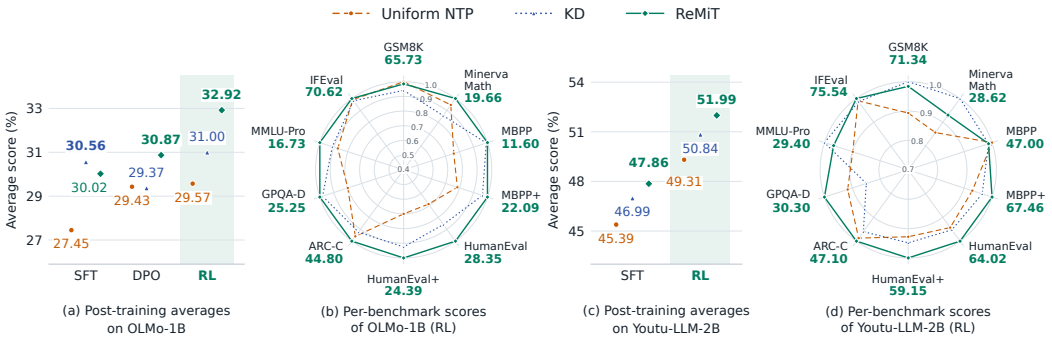



Figure 4: Post-training performance from different mid-training initializations. (a,c) Average scores (%) across ten benchmarks after each post-training stage; shading marks the final RL stage. (b,d) Task-level performance after RL. Radar labels show ReMiT scores (%).

**Training configuration.** All mid-training runs use a 50B-token budget and the high-quality corpus associated with each model family. Within each family, methods share the starting checkpoint and optimization configuration. Sequence lengths are 4,096 for OLMo-1B and 8,192 for SmolLM3-3B and Youtu-LLM-2B. ReMiT uses $\epsilon = 0.2$ with detached token weights. For downstream post-training, we use AllenAI's Open-Instruct codebase (Ivison et al., 2023) and its released datasets and training scripts for SFT, DPO, and RLVR, adapting the recipes to the evaluated model families. Within each family, the resulting mid-training initializations undergo the same subsequent training stages, using identical downstream datasets and training budgets. More details are provided in Appendix D.

**Evaluation.** Mid-trained models are evaluated on ten benchmarks covering reasoning, code, instruction following, and general capabilities, using task-specific shot counts. Post-trained models are evaluated on a ten-benchmark zero-shot suite. We report arithmetic means of the benchmark scores, with improvements expressed in percentage points. Evaluation configurations and per-task results are provided in Appendices D, and E.

### 4.2 Performance After Post-Training

We first compare the final models obtained from uniform NTP, KD, and ReMiT mid-training. Within each family, the resulting initializations undergo the same subsequent post-training recipe.

**Improved final-model performance.** ReMiT achieves the highest final average among the three methods on both evaluated families (Figure 4). On OLMo-1B, it reaches 32.92 after SFT–DPO–RL, exceeding uniform NTP and KD by 3.35 and 1.92 percentage points. On Youtu-LLM-2B, it reaches 51.99 after SFT–RL, with gains of 2.68 and 1.15 points, respectively. KD also improves over NTP at the final stage in both families. These results support the usefulness of reference guidance and show that its transfer mechanism affects final performance under the evaluated recipes.

**Benefits across downstream tasks.** Figure 4(b,d) shows that the average gains extend across multiple tasks. ReMiT outperforms uniform NTP on nine of the ten benchmarks in each model family. Full per-task scores are provided in Appendix E.

**An additional post-training recipe.** We also evaluate RL applied directly after mid-training on OLMo-1B. As shown in Figure 5, ReMiT outperforms uniform NTP in this setting as well.

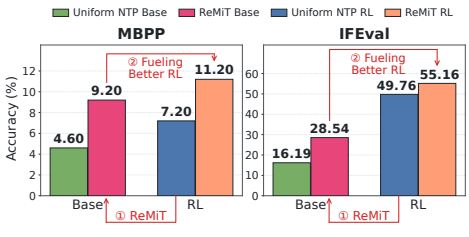



Figure 5: Direct RL after mid-training on OLMo-1B.

### 4.3 Performance Across Training Stages

**Mid-training results across three families.** Table 1 summarizes the mid-training results on OLMo-1B, SmolLM3-3B, and Youtu-LLM-2B. ReMiT consistently outperforms uniform

NTP and the evaluated MiniPLM and RHO-1 baselines across all three families, supporting the effectiveness of incorporating RL-derived feedback into corpus learning through bounded token reweighting. Unlike the evaluated RHO-1 setup, which trains a dedicated reference model, ReMiT reuses an existing RL-tuned checkpoint. This reuse avoids the additional training cost and checkpoint selection associated with constructing a separate reference model.

Table 1: Mid-training scores (%) on ten downstream benchmarks. Tasks use the evaluation protocols and shot counts specified in Appendix D. The best reported scores among the methods shown within each family are **boldfaced**; additional KD and SFT-guided comparisons appear in Table 2.

| Method | GSM8K | MATH | GPQA | BBH | IFE | HE | MBPP | TQA | ARC-C | MMLU$_P$ | Avg. |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Mid-Training on OLMo-1B | | | | | | | | | | | |
| Pre-Trained | 3.03 | 2.94 | 20.31 | 28.43 | 22.66 | 6.71 | 4.80 | 21.30 | 44.71 | 9.54 | 16.44 |
| Uniform NTP | 48.14 | 10.26 | 22.54 | 30.87 | 16.19 | 8.54 | 4.60 | 22.40 | 46.67 | 13.31 | 22.35 |
| MiniPLM | 48.45 | 9.60 | 23.21 | 30.38 | 16.79 | 7.32 | 6.80 | 23.13 | 45.31 | 13.15 | 22.41 |
| RHO-1 | 50.42 | 10.32 | **25.45** | 29.33 | 19.06 | 6.71 | 6.20 | 23.38 | 46.42 | 13.68 | 23.10 |
| ReMiT | **61.64** | **14.50** | 24.55 | **32.07** | **28.54** | **12.80** | **9.20** | **25.58** | **49.23** | **17.44** | **27.56** |
| Mid-Training on SmolLM3-3B | | | | | | | | | | | |
| Pre-Trained | 31.61 | 14.52 | **27.68** | 43.13 | 22.06 | 25.61 | 37.40 | 30.23 | **56.31** | 23.24 | 31.18 |
| Uniform NTP | **64.22** | 31.64 | 26.34 | 56.50 | 43.29 | 28.05 | 47.60 | 29.74 | 53.24 | 30.68 | 41.13 |
| MiniPLM | 60.65 | 30.84 | 26.34 | 57.10 | 41.49 | 29.88 | 48.00 | 29.13 | 53.58 | 29.52 | 40.65 |
| RHO-1 | 62.93 | 28.70 | 26.56 | 55.32 | 38.73 | 31.10 | 43.80 | 28.52 | 54.10 | 28.19 | 39.80 |
| ReMiT | 63.76 | **31.68** | 26.12 | **58.27** | **45.68** | **37.20** | **49.60** | **31.95** | 54.69 | **30.73** | **42.97** |
| Mid-Training on Youtu-LLM-2B | | | | | | | | | | | |
| Pre-Trained | 34.87 | 15.30 | 22.99 | 43.39 | 23.26 | 25.00 | 38.80 | **27.54** | 52.30 | 19.96 | 30.34 |
| Uniform NTP | 49.51 | **25.00** | 27.90 | 44.37 | 32.61 | 37.20 | 46.60 | 26.56 | 53.33 | 24.93 | 36.80 |
| MiniPLM | 51.86 | 22.76 | 26.34 | 44.20 | 33.09 | 35.98 | **47.00** | 26.56 | 53.75 | 25.34 | 36.69 |
| RHO-1 | 42.46 | 16.44 | 26.79 | 40.45 | 32.01 | 27.44 | 39.20 | 26.68 | 49.23 | 21.69 | 32.24 |
| ReMiT | **52.69** | 24.50 | **29.69** | **47.21** | **36.93** | **39.94** | **47.00** | 27.42 | **54.61** | **25.85** | **38.58** |

**The relative ordering changes across stages.** The additional comparisons in Appendix E show that KD achieves higher mid-training averages than ReMiT on both OLMo-1B and Youtu-LLM-2B. This ordering reverses after subsequent post-training: ReMiT achieves higher final averages in both families (Figure 4). On OLMo-1B, a ranking reversal also occurs within the same zero-shot post-training evaluation suite. KD leads after SFT, while ReMiT leads after DPO and RL. These results show that a higher intermediate score does not necessarily translate into better final performance under the subsequent training recipe.

### 4.4 Analysis of Reference Guidance and Weighting

**RL-tuned versus SFT references.** To examine the effect of the reference's post-training stage, we replace the RL reference with its SFT counterpart while keeping the token-weighting rule fixed. As shown in Figure 6, RL-guided ReMiT generally achieves higher accuracy during mid-training and finishes with higher scores. These results support the additional value of RL-tuned references for guiding corpus learning beyond SFT alone, under the same feedback mechanism. Full mid-training results for both evaluated model families are provided in Appendix E.

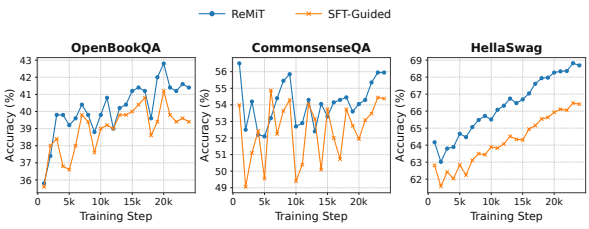



Figure 6: Comparison of an RL-tuned reference and its SFT counterpart.

**Effect of clipping.** Figure 7 compares ReMiT with a variant that removes clipping from the sigmoid-derived weights. The clipped variant maintains larger gradient norms over the displayed training interval. This observation complements the bounded-correction analysis in Section 3.2 by showing that clipping affects the realized optimization trajectory, which provide empirical evidence about the design choices.

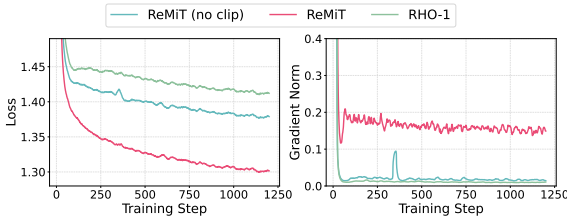



Figure 7: Clipping mechanism in ReMiT on Youtu-LLM. By bounding token-level weights, clipping keeps gradient norms within a normal range and accelerates loss reduction.

### 4.5 Reusing an Updated Reference on OLMo

**Reusing an updated RL reference.** On OLMo-1B, we reuse the first round's RL model to guide a second mid-training run, denoted ReMiT$^2$. This run restarts from the same checkpoint before mid-training, using the same corpus and training recipe.

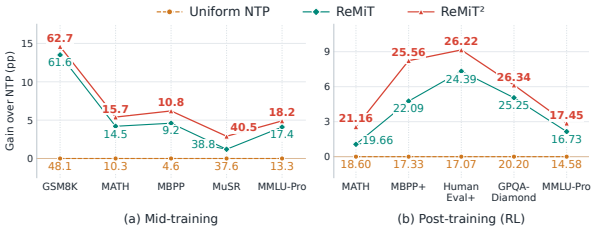



Figure 8: An additional feedback round on OLMo-1B.

As shown in Figure 8, ReMiT$^2$ improves over ReMiT on all five benchmarks shown for each stage. The gains extend to the final RL model, supporting the reuse of feedback generated within the pipeline to improve a subsequent training cycle.

## 5 Related Work

**Data Selection Strategies in Pre-Training.** Optimizing data quality is fundamental to pre-training (Xie et al., 2023; Albalak et al., 2024; Liu et al., 2024; Gu et al., 2024; Zhu et al., 2025). Existing strategies include heuristic-based and classifier-based filtering (Mann et al., 2020), as well as perplexity- or loss-based approaches (Qin et al., 2023; Wenzek et al., 2019). Recent approaches further use reference-model loss signals to select training data or tokens (Ankner et al., 2024; Lin et al., 2024). ReMiT incorporates feedback from an RL-tuned model through token-level *reweighting*, adjusting the contribution of corpus tokens while retaining their observed prediction targets.

**Token-Level Analysis.** Recent studies examine how individual tokens contribute to RL optimization. Yang et al. (2025b) show that low-probability tokens can disproportionately influence RL updates through large gradient contributions. Other work investigates the role of high-entropy tokens at pivotal decision points in RLVR (Wang et al., 2025a) and the importance of key tokens in error accumulation (Arbuzov et al., 2025). Related self-training methods also use entropy over sampled answers to adaptively weight training examples (Wang et al., 2025b). ReMiT studies token weighting during corpus-based mid-training, using reference–student prediction discrepancies as the guidance signal and assessing its utility after subsequent post-training.

## 6 Conclusion

This work investigates the coupling of pre-training and post-training, identifying the mid-training phase as a critical turning point. We introduce ReMiT, a method that leverages an RL reference to assign dynamic, token-wise weights during mid-training, guiding the base model toward pivotal reasoning tokens. Evaluations across multiple model families demonstrate that ReMiT consistently enhances both mid-training and post-training, establishing an iterative feedback loop between the base and the RL model.

### AI USE STATEMENT

We used generative AI tools to assist with the implementation and debugging of parts of the experimental and evaluation code, as well as with editing and polishing the manuscript for clarity, readability, and presentation. All AI-assisted code was reviewed and tested by the authors, and all experimental results were produced and verified through our own evaluation pipeline. The authors reviewed and revised all AI-assisted text and take full responsibility for the methodology, experiments, analysis, claims, and final content of the paper.

### REPRODUCIBILITY STATEMENT

Section 3 specifies the weighting objective and Appendix B provides the local-analysis derivation. Appendix D documents the model checkpoints, training configurations, baseline implementations, and evaluation protocols. Appendix E reports per-task results across training stages.

## REFERENCES

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

Alon Albalak, Yanai Elazar, Sang Michael Xie, Shayne Longpre, Nathan Lambert, Xinyi Wang, Niklas Muennighoff, Bairu Hou, Liangming Pan, Haewon Jeong, et al. A survey on data selection for language models. *arXiv preprint arXiv:2402.16827*, 2024.

Zachary Ankner, Cody Blakeney, Kartik K. Sreenivasan, Max Marion, Matthew L. Leavitt, and Mansheej Paul. Perplexed by perplexity: Perplexity-based data pruning with small reference models. *ArXiv*, abs/2405.20541, 2024. URL https://api.semanticscholar.org/CorpusID:270199394.

Mikhail L Arbuzov, Alexey A Shvets, and Sisong Beir. Beyond exponential decay: Rethinking error accumulation in large language models. *arXiv preprint arXiv:2505.24187*, 2025.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 7432–7439, 2020.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Yuxian Gu, Hao Zhou, Fandong Meng, Jie Zhou, and Minlie Huang. Miniplm: Knowledge distillation for pre-training language models. *arXiv preprint arXiv:2410.17215*, 2024.

Hamish Ivison, Yizhong Wang, Valentina Pyatkin, Nathan Lambert, Matthew Peters, Pradeep Dasigi, Joel Jang, David Wadden, Noah A. Smith, Iz Beltagy, and Hannaneh Hajishirzi. Camels in a changing climate: Enhancing lm adaptation with tulu 2, 2023.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. Solving quantitative reasoning problems with language models. *Advances in neural information processing systems*, 35:3843–3857, 2022.

Stephanie Lin, Jacob Hilton, and Owain Evans. Truthfulqa: Measuring how models mimic human falsehoods. *arXiv preprint arXiv:2109.07958*, 2021.

Zhenghao Lin, Zhibin Gou, Yeyun Gong, Xiao Liu, Ruochen Xu, Chen Lin, Yujiu Yang, Jian Jiao, Nan Duan, Weizhu Chen, et al. Not all tokens are what you need for pretraining. *Advances in Neural Information Processing Systems*, 37:29029–29063, 2024.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatGPT really correct? rigorous evaluation of large language models for code generation. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=1qvx610Cu7.

Yang Liu, Jiahuan Cao, Chongyu Liu, Kai Ding, and Lianwen Jin. Datasets for large language models: A comprehensive survey. *arXiv preprint arXiv:2402.18041*, 2024.

Junru Lu, Jiarui Qin, Lingfeng Qiao, Yinghui Li, Xinyi Dai, Bo Ke, Jianfeng He, Ruizhi Qiao, Di Yin, Xing Sun, et al. Youtu-llm: Unlocking the native agentic potential for lightweight large language models. *arXiv preprint arXiv:2512.24618*, 2025.

Ben Mann, Nick Ryder, Melanie Subbiah, J Kaplan, P Dhariwal, A Neelakantan, P Shyam, G Sastry, A Askell, S Agarwal, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 1(3):3, 2020.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*, 2018.

Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, et al. 2 olmo 2 furious. *arXiv preprint arXiv:2501.00656*, 2024.

Ziheng Qin, K. Wang, Zangwei Zheng, Jianyang Gu, Xiang Peng, Daquan Zhou, and Yang You. Infobatch: Lossless training speed up by unbiased dynamic data pruning. *ArXiv*, abs/2303.04947, 2023. URL https://api.semanticscholar.org/CorpusID:257427625.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9): 99–106, 2021.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. Socialiqa: Commonsense reasoning about social interactions. *arXiv preprint arXiv:1904.09728*, 2019.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

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

Team SmolLM et al. Smollm3: smol, multilingual, long-context reasoner. https://huggingface.co/blog/smollm3, 2025.

Zayne Sprague, Xi Ye, Kaj Bostrom, Swarat Chaudhuri, and Greg Durrett. Musr: Testing the limits of chain-of-thought with multistep soft reasoning. *ArXiv*, abs/2310.16049, 2023. URL https://api.semanticscholar.org/CorpusID:264439655.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, et al. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*, 2022.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. Commonsenseqa: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4149–4158, 2019.

Ellen Xiaoqing Tan, Jack Lanchantin, Shehzaad Dhuliawala, Danwei Li, Thao Nguyen, Jing Xu, Ping Yu, Ilia Kulikov, Sainbayar Sukhbaatar, Jason Weston, et al. Self-improving pretraining: using post-trained models to pretrain better models. *arXiv preprint arXiv:2601.21343*, 2026.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xionghui Chen, Jianxin Yang, Zhenru Zhang, Yuqiong Liu, An Yang, Andrew Zhao, Yang Yue, Shiji Song, Bowen Yu, Gao Huang, and Junyang Lin. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning. *ArXiv*, abs/2506.01939, 2025a. URL https://api.semanticscholar.org/CorpusID:279119146.

Xiaoxuan Wang, Yihe Deng, Mingyu Derek Ma, and Wei Wang. Entropy-based adaptive weighting for self-training. *ArXiv*, abs/2503.23913, 2025b. URL https://api.semanticscholar.org/CorpusID:277451642.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37:95266–95290, 2024.

Zengzhi Wang, Fan Zhou, Xuefeng Li, and Pengfei Liu. Octothinker: Mid-training incentivizes reinforcement learning scaling. *arXiv preprint arXiv:2506.20512*, 2025c.

Guillaume Wenzek, Marie-Anne Lachaux, Alexis Conneau, Vishrav Chaudhary, Francisco Guzmán, Armand Joulin, and Edouard Grave. Ccnet: Extracting high quality monolingual datasets from web crawl data. *arXiv preprint arXiv:1911.00359*, 2019.

Sang Michael Xie, Shibani Santurkar, Tengyu Ma, and Percy S Liang. Data selection for language models via importance resampling. *Advances in Neural Information Processing Systems*, 36:34201–34227, 2023.

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

Zhihe Yang, Xufang Luo, Zilong Wang, Dongqi Han, Zhiyuan He, Dongsheng Li, and Yunjian Xu. Do not let low-probability tokens over-dominate in rl for llms. *ArXiv*, abs/2505.12929, 2025b. URL https://api.semanticscholar.org/CorpusID:278740466.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.

Qinkai Zheng, Xiao Xia, Xu Zou, Yuxiao Dong, Shan Wang, Yufei Xue, Lei Shen, Zihan Wang, Andi Wang, Yang Li, et al. Codegeex: A pre-trained model for code generation with multilingual benchmarking on humaneval-x. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 5673–5684, 2023.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023.

Xiaoxuan Zhu, Zhouhong Gu, Baiqian Wu, Suhang Zheng, Tao Wang, Tianyu Li, Hongwei Feng, and Yanghua Xiao. Toremi: Topic-aware data reweighting for dynamic pretraining data selection. *arXiv preprint arXiv:2504.00695*, 2025.

## A Background: The Mid-Training Phase

Standard large language model (LLM) training typically comprises two stages: pre-training on large-scale corpora and post-training for alignment. Recent work highlights an intermediate stage—often termed *mid-training* or *annealing*—that bridges broad knowledge acquisition and the refinement of specialized capabilities. Mid-training is commonly associated with two shifts:

**High-quality data shift.** Whereas stable pre-training relies primarily on large volumes of general web text to build broad coverage, mid-training emphasizes data quality over quantity. As illustrated in the top panel of Figure 9, this phase increases the proportion of STEM, code, and reasoning-focused data, which can encourage the model to develop stronger competence in complex problem solving.

**Learning-rate decay.** To incorporate this higher-quality data effectively, training typically applies an accelerated learning-rate decay. As shown in the bottom panel of Figure 9, the learning rate decreases from the pre-training plateau to a near-zero value. This process can stabilize optimization and consolidate acquired behaviors, making mid-training a natural window for injecting or reinforcing reasoning-oriented capabilities before post-training.

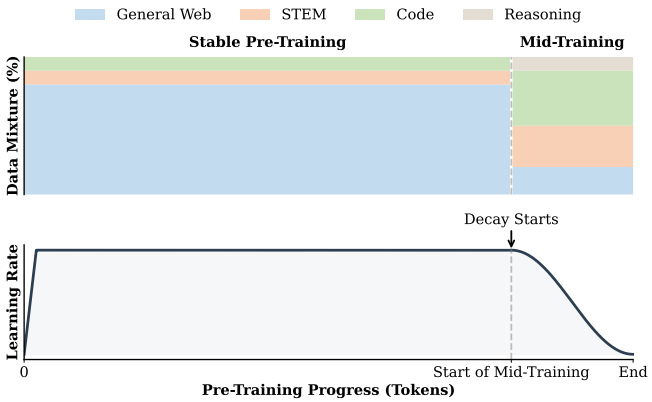



Figure 9: Overview of the pre-training pipeline. Top: During mid-training, the data distribution shifts from general web text to a higher-quality mixture enriched with code, STEM, and reasoning content. Bottom: The learning rate schedule is synchronized with this data shift, exhibiting rapid decay throughout the mid-training phase.

## B Derivation of the Local Feedback Decomposition

All quantities are evaluated at the same parameter point $\theta$, with token weights held fixed during differentiation. Expectations follow the corpus-token distribution defined in the main text.

Since $\nabla \widetilde{V}_K$ is $L$-Lipschitz in the neighborhood considered, either direction $u \in \{u_0, u_w\}$ satisfies

$$\widetilde{V}_K(\theta + \eta u) = \widetilde{V}_K(\theta) + \eta \langle \nabla \widetilde{V}_K(\theta), u \rangle + r_u, \qquad |r_u| \le \frac{L\eta^2}{2} \|u\|^2. \tag{17}$$

Subtracting the two expansions yields

$$\begin{aligned} &\widetilde{V}_K(\theta + \eta u_w) - \widetilde{V}_K(\theta + \eta u_0) \\ &\quad = \eta \left\langle \nabla \widetilde{V}_K(\theta), u_w - u_0 \right\rangle + r_{u_w} - r_{u_0} \\ &\quad = -\eta \, \mathbb{E}_z \left[ (w-1) \left\langle \nabla \widetilde{V}_K(\theta), g(z) \right\rangle \right] + r_{u_w} - r_{u_0} \\ &\quad = \eta \, \mathbb{E}_z[(w-1) a_K] + r_{u_w} - r_{u_0}. \end{aligned} \tag{18}$$



Figure 10: Token distribution divergence between the Base and RL models. Background intensity encodes the per-token log-probability margin ($\Delta \log p = \log p_{\text{RL}} - \log p_{\text{base}}$), with deeper red indicating higher confidence in the ground-truth token from the RL model.

The expectation decomposes as

$$\begin{aligned} \mathbb{E}_z[(w-1)a_K] &= \mathbb{E}_z[wa_K] - \mathbb{E}_z[a_K] \\ &= \mathrm{Cov}_z(w, a_K) + \big(\mathbb{E}_z[w] - 1\big)\mathbb{E}_z[a_K]. \end{aligned} \tag{19}$$

Setting $R_\eta = r_{u_w} - r_{u_0}$ and applying the triangle inequality gives

$$|R_\eta| \le |r_{u_w}| + |r_{u_0}| \le \frac{L\eta^2}{2}\left(\|u_w\|^2 + \|u_0\|^2\right). \tag{20}$$

These identities establish Equations equation 14 and equation 15. Condition equation 16 makes the first-order term strictly larger than the maximum remainder magnitude, ensuring a positive local difference in $\widetilde{V}_K$.




Figure 11: A comparison of token-level preferences between the RL and SFT models. The background color of each ground-truth token indicates the log-probability margin $\Delta \log p = \log p_{\text{RL}} - \log p_{\text{SFT}}$. Deeper red signifies a stronger preference from the RL model, deeper blue from the SFT model, and white denotes a negligible difference ($|\Delta \log p| < 0.1$).

# C Visualization of Log-Probability Discrepancies Between Different Models

## C.1 Base Model versus RL Model

To identify where the RL model surpasses the base model on a given sentence, Section 2.2 presents an example visualizing log-probability discrepancies. Figure 10 provides additional examples for further analysis. For each token, the background color encodes the RL-to-Base log-probability margin ($\Delta \log p$): lighter shades indicate smaller margins, whereas deeper red indicates that the RL model assigns higher likelihood to the ground-truth token. Most tokens show modest margins, while a small subset exhibits large positive margins (deep red). This pattern suggests that the RL model's superior reasoning is driven by a limited set of pivotal tokens, with the remainder largely following the base model's distribution.

## C.2 SFT Model versus RL Model

We now analyze the rationale for selecting the RL model over the SFT model as a reference during the mid-training process. A key justification lies in their distinct token-level preferences, which we now examine in greater detail. As our examples illustrate, the RL model assigns a significantly higher likelihood to discourse connectives (e.g., 'First', 'But', 'So', 'Let'). This preference, visualized by a deeper red background for the corresponding tokens, is consistent with the enhanced reasoning capabilities cultivated during reinforcement learning, confirming the RL model as the more suitable reference for ReMiT.

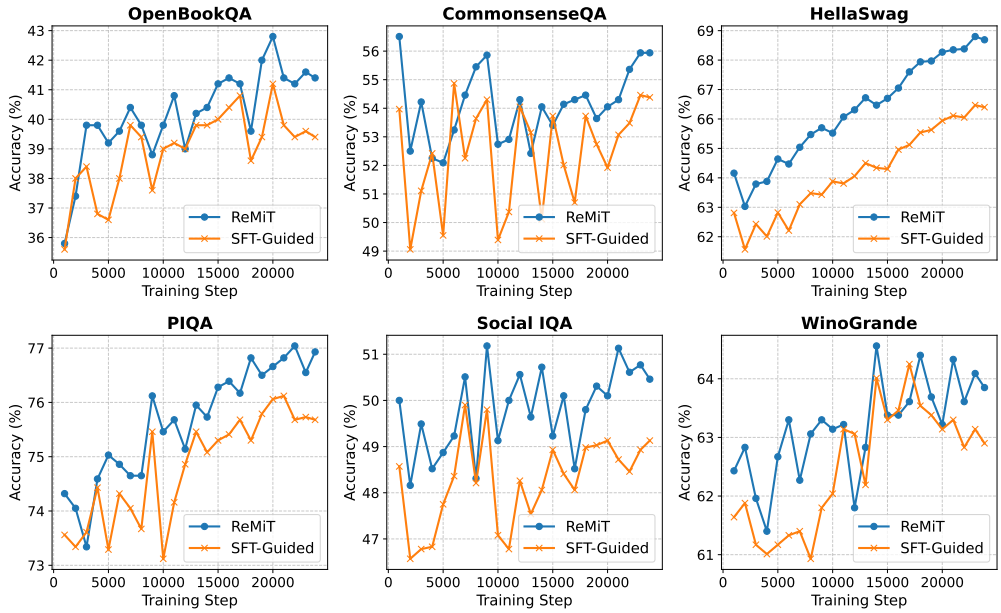



Figure 12: Comparison of downstream performance between SFT-Guided and ReMiT during mid-training on OLMo-1B. The results demonstrate that our RL-guided approach ReMiT consistently outperforms the SFT-guided baseline throughout the training trajectory.

Table 2: Extended evaluation of pre-training models across 10 widely used downstream tasks. The best scores within each model family are **boldfaced**.

| Method | GSM8K | MATH | GPQA | BBH | IFE | HE | MBPP | TQA | ARC-C | $\text{MMLU}_\text{P}$ | Avg. |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Mid-Training on OLMo-1B | | | | | | | | | | | |
| Vanilla NTP | 48.14 | 10.26 | 22.54 | 30.87 | 16.19 | 8.54 | 4.60 | 22.40 | 46.67 | 13.31 | 22.35 |
| SFT-Guided | 50.34 | 12.16 | 23.44 | 31.15 | 13.07 | 10.98 | 9.00 | 24.11 | 47.70 | 15.23 | 23.72 |
| KD | 61.26 | **15.88** | 23.88 | **32.11** | **45.56** | **13.41** | **13.20** | **25.83** | **49.23** | **17.45** | **29.78** |
| ReMiT | **61.64** | 14.50 | **24.55** | 32.07 | 28.54 | 12.80 | 9.20 | 25.58 | **49.23** | 17.44 | 27.56 |
| Mid-Training on Youtu-LLM-2B | | | | | | | | | | | |
| Vanilla NTP | 49.51 | **25.00** | 27.90 | 44.37 | 32.61 | 37.20 | 46.60 | 26.56 | 53.33 | 24.93 | 36.80 |
| SFT-Guided | 47.54 | 19.84 | **30.13** | 44.60 | 32.73 | 32.93 | 43.40 | 25.46 | 52.65 | 24.42 | 35.37 |
| KD | **60.42** | 23.86 | 25.67 | **50.71** | **38.61** | 32.93 | 45.20 | **29.38** | 52.13 | **27.77** | **38.67** |
| ReMiT | 52.69 | 24.50 | 29.69 | 47.21 | 36.93 | **39.94** | **47.00** | 27.42 | **54.61** | 25.85 | 38.58 |

# D Experimental Configuration

## D.1 Mid-Training Corpus

For the data setup, we utilize the official mid-training mixes associated with each model family. These corpora share a similar high-quality composition, blending general web text with domain-specific data such as code, mathematics, and STEM reasoning. For all three models, the mid-training phase is conducted on a total of 50B tokens. The sequence length is set to 4,096 for OLMo and 8,192 for both SmolLM3 and Youtu-LLM, consistent with their original configurations.

## D.2 Optimization Hyperparameters

We employ a cosine decay schedule down to a minimum learning rate of 1e-7. The specific configurations are:

- **OLMo-1B:** Maximum learning rate 3e-4, Global batch size 2M tokens.

Table 3: Zero-shot accuracy of post-trained models on OLMo-1B across 10 widely used downstream tasks. The best scores are **boldfaced**.

| Method | GSM8K | MATH | MBPP | MBPP+ | HE | HE+ | ARC-C | $GPQA_{DM}$ | $MMLU_P$ | IFE | Avg. |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Stage: SFT | | | | | | | | | | | |
| Vanilla NTP | 42.61 | 12.28 | 16.20 | 25.66 | 21.95 | 18.90 | 40.96 | 24.75 | 14.34 | 56.83 | 27.45 |
| KD | 41.55 | 15.22 | **21.20** | **33.33** | **30.49** | **26.22** | 38.91 | 25.25 | **15.41** | 58.03 | **30.56** |
| ReMiT | **47.00** | **16.54** | 19.60 | 29.10 | 26.22 | 21.95 | **41.38** | **25.76** | 13.56 | **59.11** | 30.02 |
| Stage: DPO | | | | | | | | | | | |
| Vanilla NTP | **57.70** | 15.10 | 9.20 | 18.78 | 24.39 | 21.95 | 43.43 | 21.72 | 14.61 | 67.39 | 29.43 |
| KD | 49.66 | 15.96 | **12.80** | **22.75** | 22.56 | 20.73 | 39.25 | **26.34** | **15.80** | 67.87 | 29.37 |
| ReMiT | 55.04 | **17.96** | 12.40 | 21.69 | **25.61** | **22.56** | **45.22** | 23.23 | 15.78 | **69.18** | **30.87** |
| Stage: RL | | | | | | | | | | | |
| Vanilla NTP | **66.72** | 18.60 | 8.80 | 17.33 | 19.51 | 17.07 | 43.00 | 20.20 | 14.58 | 69.90 | 29.57 |
| KD | 62.77 | 16.84 | 11.50 | 21.30 | 24.39 | 22.56 | 41.38 | 24.75 | 15.24 | 69.30 | 31.00 |
| ReMiT | 65.73 | **19.66** | **11.60** | **22.09** | **28.35** | **24.39** | **44.80** | **25.25** | **16.73** | **70.62** | **32.92** |

Table 4: Zero-shot accuracy of post-trained models on Youtu-LLM-2B across 10 widely used downstream tasks. The best scores are **boldfaced**.

| Method | GSM8K | MATH | MBPP | MBPP+ | HE | HE+ | ARC-C | $GPQA_{DM}$ | $MMLU_P$ | IFE | Avg. |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Stage: SFT | | | | | | | | | | | |
| Vanilla NTP | 57.32 | 22.46 | 46.60 | 62.17 | 56.71 | 50.61 | 43.77 | 24.75 | 23.66 | 65.83 | 45.39 |
| KD | 63.38 | **27.28** | 45.00 | 61.90 | **57.32** | 51.22 | 43.00 | 27.27 | **27.02** | **66.55** | 46.99 |
| ReMiT | **64.06** | 22.84 | **49.40** | **65.34** | **57.32** | **53.66** | **45.22** | **29.80** | 25.41 | 65.59 | **47.86** |
| Stage: RL | | | | | | | | | | | |
| Vanilla NTP | 64.75 | 26.36 | **47.60** | 62.70 | 60.37 | 54.88 | 46.50 | 27.78 | 27.32 | 74.82 | 49.31 |
| KD | **72.48** | **30.76** | 47.00 | 65.08 | 60.98 | 56.10 | 45.22 | 25.76 | **30.40** | 74.58 | 50.84 |
| ReMiT | 71.34 | 28.62 | 47.00 | **67.46** | **64.02** | **59.15** | **47.10** | **30.30** | 29.40 | **75.54** | **51.99** |

- **SmolLM3-3B:** Maximum learning rate 2e-4, Global batch size 2M tokens.
- **Youtu-LLM-2B:** Maximum learning rate 2e-4, Global batch size 64M tokens.

These settings are consistent with the configurations used during the respective stable pre-training phases of each model.

### D.3 Post-Training Details

We use Open-Instruct for downstream SFT, DPO, and reinforcement learning with verifiable rewards. SFT uses 'allenai/tulu-3-sft-olmo-2-mixture-0225', and DPO uses 'allenai/olmo-2-0425-1b-preference-mix'. The RL stage uses 'allenai/RLVR-GSM-MATH-IF-Mixed-Constraints', which combines mathematical problems and prompts with verifiable instruction-following constraints. The policy optimizer is GRPO, with rewards produced by the corresponding task verifiers. OLMo-1B follows SFT–DPO–RL, whereas Youtu-LLM-2B follows SFT–RL. We also evaluate direct RL from the OLMo-1B mid-training checkpoint. Within each family, data and stage-specific training budgets are identical across compared initializations.

### D.4 Evaluation of Pre-trained Base LMs

We comprehensively evaluate the pre-trained base models by measuring their few-shot performance across a suite of widely used downstream benchmarks, including: GSM8K (Cobbe et al. (2021); 8-shot), Minerva Math (Lewkowycz et al. (2022); 4-shot), GPQA (Rein et al. (2024); 5-shot), BigBenchHard (BBH; Suzgun et al. (2022); 3-shot CoT), IFEval (IFE; Zhou et al. (2023) 3-shot), HumanEval (HE; Zheng et al. (2023); 0-shot), MBPP (Austin et al. (2021); 3-shot), TruthfulQA (TQA; Lin et al. (2021); 3-shot), ARC-Challenge (Clark et al. (2018); 25-shot), and $MMLU_{Pro}$ (Wang et al. (2024); 5-shot). Unless otherwise specified, we report Pass@1 for both MBPP and HumanEval. In extended experiments, we also evaluate additional benchmarks such as HellaSwag (Zellers et al., 2019), PIQA (Bisk et al., 2020),

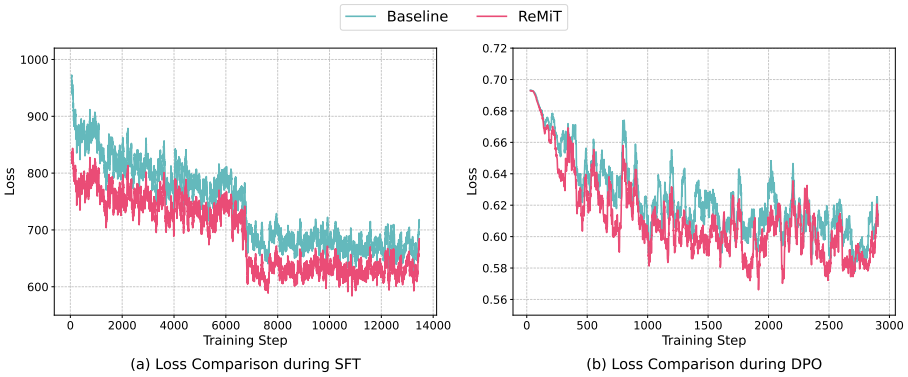



Figure 13: Loss trajectories during SFT and DPO on OLMo-1B.

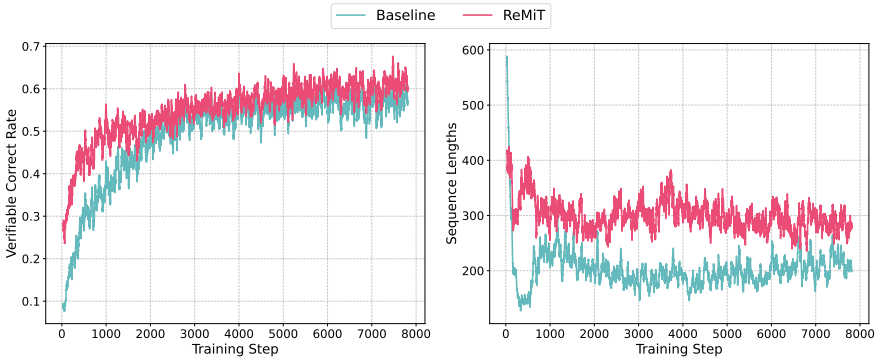



Figure 14: Performance comparison at the reinforcement learning stage on OLMo-1B.

Social IQA Sap et al. (2019), WinoGrande Sakaguchi et al. (2021), OpenBookQA Mihaylov et al. (2018), CommensenseQA Talmor et al. (2019) and MuSR (Sprague et al., 2023).

### D.5 Evaluation of Post-trained LMs

Given that the post-trained models already exhibit instruction-following ability, we evaluate their zero-shot accuracy. We report results on 10 widely used benchmarks: GSM8K (Cobbe et al., 2021), Minerva Math (Lewkowycz et al., 2022), MBPP (Austin et al., 2021), MBPP+ (Liu et al., 2023), HumanEval (HE; Zheng et al. (2023)), HumanEval+ (HE+; Liu et al. (2023)), ARC-Challenge (Clark et al., 2018), $\text{GPQA}_{\text{Diamond}}$ ($\text{GPQA}_{\text{DM}}$; Rein et al. (2024)), $\text{MMLU}_{\text{Pro}}$ (Wang et al., 2024), IFEval (IFE; Zhou et al. (2023)).

## E Extended Analysis and Results

### E.1 Pre-Training Results

Table 2 extends the mid-training comparisons with KD and SFT-guided weighting on OLMo-1B and Youtu-LLM-2B.

**Reference comparison.** With the weighting rule fixed, RL-guided ReMiT achieves higher mid-training averages than SFT-guided weighting on both families: 27.56 versus 23.72 on OLMo-1B and 38.58 versus 35.37 on Youtu-LLM-2B. SFT guidance itself improves over uniform NTP on OLMo-1B but falls below it on Youtu-LLM-2B. Figure 12 provides additional OLMo-1B trajectories, with higher final scores for ReMiT on the displayed tasks despite crossings at intermediate checkpoints. These comparisons support the evaluated RL ref-

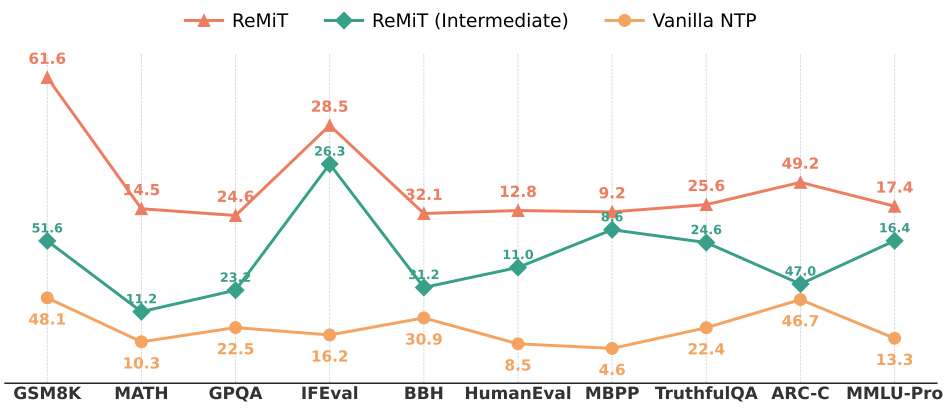



Figure 15: Impact of RL reference quality on mid-training performance across 10 downstream tasks. We compare ReMiT (using a fully converged RL model) against ReMiT (Intermediate) which uses a checkpoint trained for only 10% of RL steps.

erences under the same weighting mechanism; the token visualizations in Appendix C provide qualitative context.

**Knowledge distillation.** KD achieves mid-training averages of 29.78 on OLMo-1B and 38.67 on Youtu-LLM-2B, compared with 27.56 and 38.58 for ReMiT. The final post-training ordering is different, as reported in Section 4. This change motivates evaluation after the intended training pipeline; it does not, by itself, identify overfitting or reduced plasticity as the cause of KD's relative performance.

### E.2 Post-Training Results

Tables 3 and 4 report task-level results after each evaluated post-training stage. ReMiT achieves the highest final average in both families, while KD also remains above uniform NTP at the final stage. On OLMo-1B, KD leads after SFT and ReMiT leads after DPO and RL. These results distinguish the immediate value of a mid-training objective from the final performance reached under subsequent post-training.

Figures 13 and 14 provide training diagnostics for OLMo-1B. ReMiT generally exhibits lower logged SFT and DPO losses and higher RL verifiable correct rates over the displayed intervals. Generated sequences are also longer during much of RL training. These diagnostics describe the optimization trajectories; final task performance is assessed through the evaluation tables.

## F Discussion

### F.1 Impact of RL Reference Quality

We study how the quality of the RL reference affects mid-training by using an intermediate RL checkpoint (trained for only 10% of the total RL steps) as a weaker reference, denoted ReMiT (Intermediate). We compare this variant with standard ReMiT (using the fully converged RL model) and the Uniform NTP baseline. Results are shown in Figure 15.

**Sensitivity to signal quality.** We observe a clear performance ordering: ReMiT $>$ ReMiT (Intermediate) $>$ Uniform NTP. The observed positive correlation between reference quality and downstream performance confirms that ReMiT effectively leverages the specific reasoning priors embedded in the reference signal.

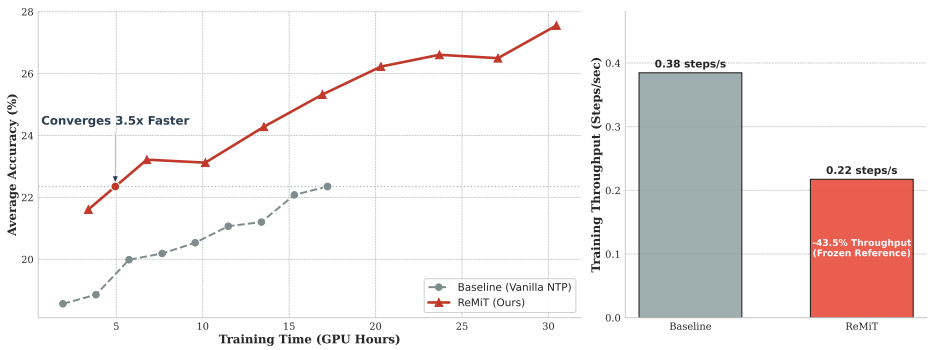



Figure 16: Training efficiency analysis of OLMo-1B on the same 50B mid-training corpus. (a) Wall-clock Convergence: Despite the computational overhead of the reference model, ReMiT demonstrates superior efficiency, converging 3.3x faster than the baseline in terms of GPU hours. (b) Throughput Comparison: While ReMiT incurs a reduction in training throughput (steps/sec) due to the additional forward pass, this cost is effectively amortized by its rapid convergence rate.

**Robustness and safety boundaries.** Despite being trained for only 10% of the RL steps, ReMiT (Intermediate) consistently outperforms Uniform NTP, indicating robustness to imperfect guidance. Moreover, the clipping mechanism in Equation (5) provides an explicit safeguard by limiting extreme token weights, thereby reducing the risk of overfitting to noise or bias in a weaker reference.

**Implications for the in-pipeline strategy.** These findings further support the in-pipeline design (Figure 3): the observed gains can be obtained using an RL reference produced within the same pipeline and of the same parameter scale, yielding a reliable and consistent signal without relying on larger, external teachers. This reinforces our central contribution—an iterative evolution cycle between pre-training and post-training. By feeding post-training improvements back into mid-training, ReMiT forms a self-reinforcing flywheel that enables continual progress while avoiding the additional computational cost of training separate, external reference models.

### F.2 Computational Efficiency and Overhead Analysis

A potential concern with reference-guided approaches is the computational overhead introduced by the additional forward pass of the reference model. To address this, we analyze the trade-off between training throughput and convergence speed, demonstrating that ReMiT offers a superior compute-performance ratio compared to standard next token prediction.

**Time-to-Convergence Efficiency.** While Figure 1 establishes ReMiT's sample efficiency—showing it reaches baseline performance using significantly fewer tokens (6× faster in token count) —Figure 15 provides a critical wall-clock analysis that accounts for real-world computational costs. As shown in Figure 15(a), ReMiT achieves a 3.5x speedup in time-to-convergence compared to the Uniform NTP baseline, measured in GPU hours. This result is significant because it confirms that the gains in sample efficiency vastly outweigh the reduction in training throughput.

**Overhead vs. Gain.** Figure 15(b) explicitly quantifies the computational cost: ReMiT incurs a 43% reduction in throughput (steps/sec) due to the inference overhead of the frozen RL reference. However, this cost is effectively amortized by the method's rapid learning trajectory. Crucially, the performance ceiling achieved by ReMiT in Figure 15(a) is not merely a result of accelerated convergence; the curve shows that ReMiT reaches accuracy

levels that the Uniform NTP baseline struggles to attain even with extended training. This indicates that ReMiT provides a qualitative improvement in reasoning capabilities that cannot be replicated simply by training a standard model for longer periods or with more tokens. Moreover, given that the mid-training stage typically occupies a short window at the end of pre-training, this temporary overhead does not pose a barrier for scaling to larger models. Future research could investigate strategies to mitigate this overhead, for instance, through offline weight computation or asynchronous scoring mechanisms.