CoDistill-GRPO: A Co-Distillation Recipe for Efficient Group Relative Policy Optimization

Anonymous Authors[1]

Abstract

Group Relative Policy Optimization (GRPO) has emerged as a powerful algorithm for improving reasoning in language models, but often fails to improve small models due to sparse rewards on difficult tasks. Existing works attempt to mitigate this issue by leveraging a larger language model as a frozen oracle, either to provide hints for rollouts or for knowledge distillation (KD). However, this assumes the existence of such an oracle, and training one can significantly increase total training time. In this work, we propose CoDistill-GRPO, a co-distillation algorithm that simultaneously trains a large and a small model by maximizing the GRPO objective. The two models learn from each other: the small model uses an on-policy KD reward to learn from the large model's distribution, while the large model is updated using rollouts generated by the small model, reducing the computational overhead of rollout generation. We show that CoDistill-GRPO substantially improves small-model performance over GRPO on mathematical benchmarks across both Qwen and Llama models, with an accuracy increase of 6.0 percentage points on the Minerva dataset for Qwen2.5-Math-1.5B. Interestingly, we also show that the large model trained with CoDistill-GRPO can nearly match standard GRPO performance despite training on small model rollouts. This highlights CoDistill-GRPO as a potential cost-effective alternative to GRPO for larger models.

1. Introduction

Reinforcement learning with verifiable rewards (RLVR) (Cobbe et al., 2021; Wen et al., 2026; Lambert et al., 2025; Yue et al., 2025; DeepSeek-AI et al., 2025) has emerged as a canonical paradigm for aligning large language models (LLMs) to efficiently solve downstream reasoning tasks. The objective of RLVR is to maximize rewards derived from rule-based or deterministic verifiers (e.g., correct or incorrect binary rewards), and can generally be expressed in the following mathematical form:

$$\max_{\theta} \mathbb{E}_{q\sim P(Q)}\Big[\underbrace{\mathbb{E}_{o\sim\pi_\theta}\left[r(q,o)\right]}_{\text{reward maximization}} - \beta \cdot \underbrace{\mathbb{D}_{\text{KL}}[\pi_\theta \| \pi_{\text{ref}}]}_{\text{KL regularization}}\Big], \quad (1)$$

where $o \sim \pi_\theta$ denotes an output sampled from some policy π_θ parameterized by θ for some query $q \sim P(Q)$, and $\beta \geq 0$ denotes a regularization parameter such that the policy π_θ does not deviate too much from a reference policy π_{ref}.

Since the work of DeepSeekMath (Shao et al., 2024), Group Relative Policy Optimization (GRPO) has received widespread attention, particularly for mathematical reasoning tasks and hence with verifiable rewards. GRPO is particularly appealing for its simple yet effective objective, as it does not require training a separate value model for advantage computation, as is done in algorithms such as Proximal Policy Optimization (PPO) (Ouyang et al., 2022a). This sparked a wide range of efforts aimed at improving GRPO in terms of performance and enhancing its training stability (Xiao et al., 2025; Liu et al., 2025; Zheng et al., 2025a; Yu et al., 2025; Shrivastava et al., 2025; Xu et al., 2025b).

On the other hand, recent research has identified the limitations of standard GRPO to effectively post-train small language models (Zhang et al., 2025b; Xu et al., 2025a). The main challenge is the "learning cliff" (Zhang et al., 2026): on difficult reasoning tasks: GRPO on smaller models yields sparse rewards due to their limited capacity, leading to slow or stagnant learning compared to larger models (see Figure 1). To address this issue, most existing works rely on these larger, more powerful models (or ground truth solutions) to either provide hints for rollouts (Zhang et al., 2025b) or to provide dense rewards through knowledge distillation (KD) (Xu et al., 2025a; Agarwal et al., 2024; Lu & Lab, 2025). However, these methods typically assume the existence of a frozen large-model oracle throughout the training process. While one could first train a larger model and subsequently apply these techniques, such a sequential pipeline substantially increases total training time. More-

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

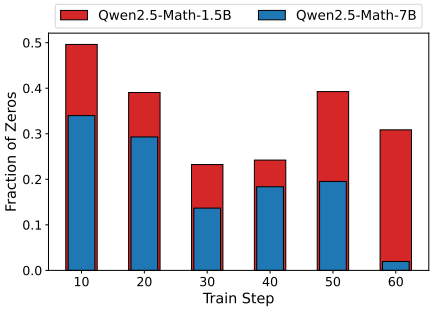

Figure 1. Fraction of zero accuracy rewards on the training iterations on the MATH dataset (Hendrycks et al., 2021) for the Qwen2.5-Math-1.5B and Qwen2.5-Math-7B models. The smaller model has a higher fraction of zero rewards, and hence learns slower across training iterations.

over, training a larger model first could widen the discrepancy between the small and large models, which has been shown to lead to poor learning when teacher and student models differ greatly in capacity and performance (Huang et al., 2022; Cho & Hariharan, 2019; Mirzadeh et al., 2019).

In this work, we mitigate these issues with CoDistill-GRPO, a co-distillation recipe that simultaneously trains a large and a small model by maximizing carefully designed GRPO objectives. The two models learn from each other: the small model uses an on-policy KD reward to learn from the large model's distribution, while the large model is updated using rollouts generated solely by the small model with importance reweighting. Switching the generation phase to use only the smaller model significantly reduces training overhead, since rollout generation is the primary bottleneck in GRPO (Liu et al., 2026; Zheng et al., 2025b). Interestingly, we show that the on-policy KD reward is largely sufficient for the small model to produce outputs suitable for updating both models. To further improve the quality and diversity of small model rollouts, we take advantage of the fact that small model rollouts are inexpensive and increase the number of small model generations. Following (Xu et al., 2025b), the set of generations is then downsampled based on a combined accuracy and KD score to obtain enhanced relative advantage signals. Across both Qwen and Llama model families, we demonstrate that CoDistill-GRPO substantially improves small model performance on mathematical reasoning benchmarks, including Minerva, MATH500, AMC2024, and OlympiadBench. For example, we show that Qwen2.5-Math-1.5B can achieve an accuracy of 32% on the Minerva dataset, a 6 percentage points improvement over GRPO. Interestingly, we also show that the large model produced by CoDistill-GRPO nearly matches standard GRPO performance despite training on small model rollouts, achieving an 18% improvement in training speed — highlighting that CoDistill-GRPO can potentially be used as a cost-effective alternative for larger models.

2. Background

Following the form in Equation (1), the goal of GRPO is to find a policy that maximizes the following objective function:

$$\mathcal{L}_{\text{GRPO}}(\theta) = \mathbb{E}_{q\sim P(Q),\{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|q)} \frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\min\left\{\frac{\zeta_{i,t}(\theta)}{\zeta_{i,t}(\theta_{\text{old}})}\hat{A}_i,\ \text{clip}_\epsilon\left(\frac{\zeta_{i,t}(\theta)}{\zeta_{i,t}(\theta_{\text{old}})}\right)\hat{A}_i\right\} - \beta\cdot\mathbb{D}_{\text{KL}}\big[\pi_\theta(\cdot|q)\|\pi_{\text{ref}}(\cdot|q)\big], \quad (2)$$

where $\zeta_{i,t}(\theta) := \pi_\theta(o_{i,t}|q, o_{i,<t})$; π_θ and $\pi_{\theta_{\text{old}}}$ are the current and old policy models, respectively; q is a question sampled from the dataset; and each o_i is an output (or rollout) generated from the old policy $\pi_{\theta_{\text{old}}}$. Similar to the PPO objective (Ouyang et al., 2022b), there is a clipping operator introduced for the importance sampling ratio, where $\text{clip}_\epsilon(x)$ outputs x if $1-\epsilon \le x \le 1+\epsilon$, $1-\epsilon$ if $x < 1-\epsilon$, and $1+\epsilon$ if $x > 1+\epsilon$ for some $\epsilon \ge 0$. However, different from PPO, the advantages $\hat{A}_i$ are computed by taking the normalized rewards across all generated G outputs, as opposed to using generalized advantage estimation (GAE). Specifically, the advantages are computed as follows:

$$\hat{A}_i = \frac{r_i - \text{mean}(\mathbf{r})}{\text{std}(\mathbf{r})}, \quad (3)$$

where r_i is the reward for the output o_i and $\mathbf{r} = \{r_1, \ldots, r_G\}$ are the aggregated rewards across all G rollouts. The KL divergence term is typically estimated using the following unbiased estimator:

$$\widehat{\mathbb{D}}_{\text{KL}}[\pi_\theta(o_i|q)\,\|\,\pi_{\text{ref}}(o_i|q)] = \frac{\pi_{\text{ref}}(o_i|q)}{\pi_\theta(o_i|q)} - \log\left(\frac{\pi_{\text{ref}}(o_i|q)}{\pi_\theta(o_i|q)}\right) - 1. \quad (4)$$

3. CoDistill-GRPO: An Efficient Co-Distillation Recipe for GRPO

In this section, we present each component of CoDistill-GRPO by following the flowchart in Figure 2. Throughout this section, we use θ and ϕ to denote the parameters of the large and small models, respectively. In Section 3.1, we present the loss function for the small model $\mathcal{L}_{\texttt{Small}}(\phi)$ and in Section 3.2, we present the loss function for the large model $\mathcal{L}_{\texttt{Large}}(\theta)$. Then, in Section 3.3, we present the downsampling mechanism. Lastly, in Section 3.4, we discuss theoretical guarantees regarding the small model loss $\mathcal{L}_{\texttt{Small}}(\phi)$ and its use of the on-policy KD reward.

3.1. GRPO Objective for the Small Model

We begin with the GRPO objective function for the small model, $\mathcal{L}_{\texttt{Small}}(\phi)$. Following the flowchart in Figure 2, the

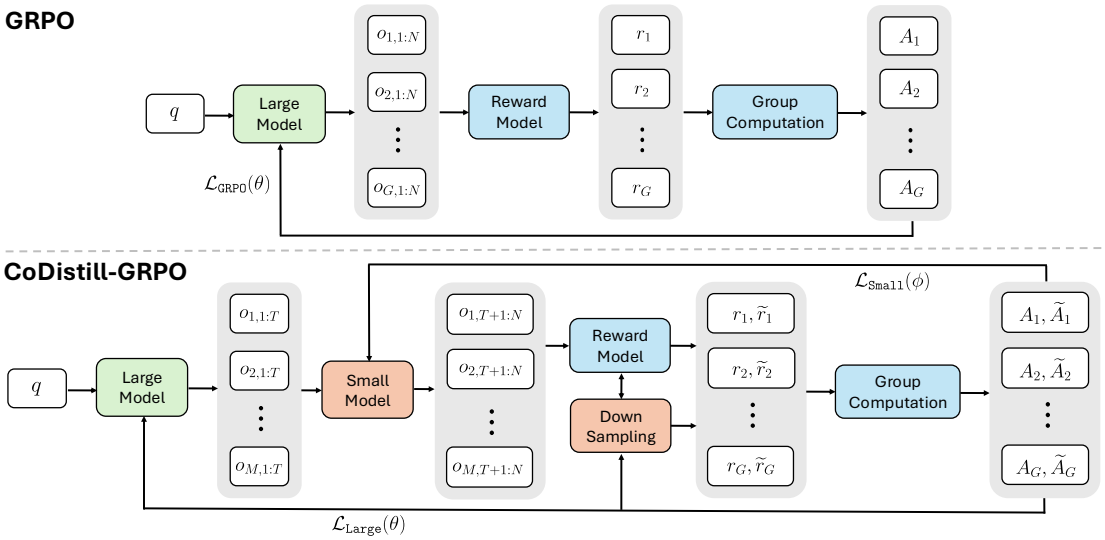

Figure 2. Flowchart of CoDistill-GRPO. CoDistill-GRPO jointly trains two models, reducing training cost by using rollouts generated by the small model. The large model provides hint tokens which the small model completes, and the resulting sequences are scored by a reward model. Because small-model rollouts are inexpensive, we generate many candidates ($M \gg G$) and apply downsampling. The large model's log probabilities are used both to compute a KD-based reward for the small model and to score rollouts for downsampling. The small model is updated using advantages $\widetilde{A}_i$ (see Equation (6)), while the large model is updated using the original A_i.

large model generates an initial $T \geq 0$ tokens as a hint for which the small model completes, which are then used to update both models. For simplicity, let us initially assume that $T = 0$, i.e., the small model generates the entire set of outputs from some model $\pi_{\phi_{\text{old}}}$. The objective $\mathcal{L}_{\texttt{Small}}(\phi)$ has a very similar structure to Equation (2), where the differences are highlighted in red:

$$\mathcal{L}_{\text{Small}}(\phi) = \mathbb{E}_{q\sim P(Q),\{o_i\}_{i=1}^{G}\sim\pi_{\phi_{\text{old}}}(\cdot|q)} \frac{1}{G}\sum_{i=1}^{G}\frac{1}{N}\sum_{t=1}^{N}\min\left\{\frac{\zeta_{i,t}(\phi)}{\zeta_{i,t}(\phi_{\text{old}})}\widetilde{A}_i, \text{clip}_\epsilon\left(\frac{\zeta_{i,t}(\phi)}{\zeta_{i,t}(\phi_{\text{old}})}\right)\widetilde{A}_i\right\}, \tag{5}$$

$$\widetilde{r}_i := \widetilde{r}(q, o_i) \tag{6}$$
$$= \underbrace{r(q, o_i)}_{\text{original reward}} + \underbrace{\alpha \cdot \frac{1}{N}\sum_{t=1}^{N}\log\left(\frac{\pi_\theta(o_{i,t}|q, o_{i,<t})}{\pi_\phi(o_{i,t}|q, o_{i,<t})}\right)}_{\text{on-policy KD reward}},$$

$$\widetilde{A}_i = \widetilde{r}_i - \text{mean}\left(\widetilde{\mathbf{r}}\right), \tag{7}$$

$\alpha > 0$ is a hyperparameter and $N > 0$ denotes the number of maximum completion tokens. The only difference lies in the rewards (and hence advantages): in addition to the original reward (e.g., accuracy for mathematical problems), the small model considers an additional reward inspired by on-policy KD (Agarwal et al., 2024). This additional reward incentivizes the small model to generate outputs that the large model would also generate, while penalizing those that the large model would not. This reward drives the probability distribution of the small model closer to that of the large model, achieving two goals: (i) improving the performance of the small model and (ii) promoting the use of the small model's rollouts to update the large model. Note that we also do not divide by the standard deviation for the advantages, following (Liu et al., 2025).

The derivation of the effective reward $\widetilde{r}_i$ is also intuitive: if we consider the GRPO objective in Equation (2) as a "reward maximization" problem with a KL regularization term between the small and large model, then we have the following:

$$\max_\phi \mathbb{E}_{q\sim P(Q)}\left[\mathbb{E}_{o_i\sim\pi_\phi}\left[r(q, o_i)\right] - \alpha \cdot \mathbb{D}_{\text{KL}}[\pi_\phi(\cdot|q)\|\pi_\theta(\cdot|q)]\right]$$
$$= \max_\phi \mathbb{E}_{q\sim P(Q),o_i\sim\pi_\phi}\left[r(q, o_i) - \alpha \cdot \log\left(\frac{\pi_\phi(o_i|q)}{\pi_\theta(o_i|q)}\right)\right].$$

Then, by averaging over all N tokens, we obtain the effective reward in Equation (6) for each o_i. Following (Liu et al., 2025), we remove the standard deviation when computing advantages and normalize with N instead of $|o_i|$, which has been shown to mitigate a potential length bias.

Discussion. There are a few design choices here that are worthy of discussion. First, we use a different unbiased estimate of the KL divergence than in Equation (4) for the reward computation, as we find that using the estimator in Equation (4) makes training unstable. Second, notice that instead of incorporating the on-policy KD term into the reward, we could have directly added it to the objective function as done in Equation (2). However, we find that

incorporating the KD term into the effective reward generally yields better performance even for the same values of α. Finally, we do not include a regularizer from a reference policy, i.e., we set $\beta = 0$ in Equation (2). Prior work has shown that omitting this term often leads to better performance (Liu et al., 2025; Hu et al., 2025), and we also aim for the small model to deviate from its initialization so that it can achieve sufficient improvement through GRPO.

Importance Reweighting for Hint Tokens. Now, let us consider the case when $T > 0$. When the large model presents an initial T tokens as a hint, we have two options for importance sampling: (i) perform importance sampling on all tokens respectively as such:

$$\underbrace{\sum_{t=1}^{T} \min\left(\frac{\zeta_{i,t}(\phi)}{\zeta_{i,t}(\theta_{\text{old}})}\widetilde{A}_{i,t}, \text{clip}_{\epsilon}\left(\frac{\zeta_{i,t}(\phi)}{\zeta_{i,t}(\theta_{\text{old}})}\right)\widetilde{A}_{i,t}\right)}_{\text{importance sampling on initial tokens from large model}} + \underbrace{\sum_{t=T+1}^{|o_i|} \min\left(\frac{\zeta_{i,t}(\phi)}{\zeta_{i,t}(\phi_{\text{old}})}\widetilde{A}_{i,t}, \text{clip}_{\epsilon}\left(\frac{\zeta_{i,t}(\phi)}{\zeta_{i,t}(\phi_{\text{old}})}\right)\widetilde{A}_{i,t}\right)}_{\text{importance sampling on completed tokens from small model}}, \tag{8}$$

or (ii) ignore importance sampling on the hint tokens and perform importance sampling solely on those produced by the small model, which would amount to just using Equation (8). The second option is what is similarly used in the objective function of BREAD (Zhang et al., 2025b). However, we find that the second option generally makes training unstable, while the first option does not perform as well as treating these initial tokens as if they were generated by the small model. Hence, even for $T > 0$, we use the objective in Equation (5). Furthermore, we also observe that providing these initial tokens as hints for all iterations leads to instability. We hypothesize that this occurs because the small model often fails to complete rollouts correctly, which produces large gradients. For this reason, we provide hints only during the early iterations as a warm start for the small model. Taken together, these two design choices create a clear distinction between our algorithm and BREAD.

3.2. GRPO Objective for the Large Model

The GRPO objective for the large model $\mathcal{L}_{\texttt{Large}}(\theta)$ is straightforward: it must account for the fact that rollouts are generated by the small model $\pi_{\phi_{\text{old}}}$. We handle this by treating the problem as an off-policy RL problem and applying importance sampling with respect to $\pi_{\phi_{\text{old}}}$, which leads to the following objective:

$$\mathcal{L}_{\text{Large}}(\theta) = \mathbb{E}_{q\sim P(Q),\{o_i\}_{i=1}^{G}\sim\pi_{\phi_{\text{old}}}(\cdot|q)} \tag{9}$$

$$\frac{1}{G}\sum_{i=1}^{G}\frac{1}{N}\sum_{t=1}^{N}\min\left\{\frac{\zeta_{i,t}(\theta)}{\zeta_{i,t}(\phi_{\text{old}})}\hat{A}_i, \text{clip}_{\epsilon}\left(\frac{\zeta_{i,t}(\theta)}{\zeta_{i,t}(\phi_{\text{old}})}\right)\hat{A}_i\right\}. \tag{10}$$

Equation (10) has the same structure as the GRPO objective in Equation (2), but now accounts for the rollouts generated via the small model. Similar to Section 3.1, we omit the regularizer with the reference model and normalize by N.

3.3. Downsampling

Lastly, we discuss the downsampling block in Figure 2. The main idea is as follows: since all rollouts are generated using the more inexpensive small model, we can produce $M > G$ rollouts and then select a subset of G rollouts that will likely lead to the most informative updates. Xu et al. (2025b) show that choosing the rollouts with the top and bottom $G/2$ rewards yields better performance than selecting the top-G rollouts with the highest rewards. Intuitively, this maximizes reward variance and exposes the model to both strong and weak trajectories. Similarly, CoDistill-GRPO selects the top and bottom $G/2$ rollouts based on the effective reward in Equation (6), which incorporates the KD score, as these selected rollouts are also used to update the large model.

3.4. Theoretical Results

We further establish the theoretical soundness of Co-Distill GRPO with respect to the small model loss $\mathcal{L}_{\texttt{Small}}(\phi)$. First, we prove that even with an effective reward incorporating the on-policy KD term, the gradient estimate remains unbiased. Second, we show that the gradient of $\mathcal{L}_{\texttt{Small}}(\phi)$ decomposes into two distinct directions: one optimizing the original reward and another tracking the large model. This formalizes how setting $\alpha > 0$ guides the small model toward the large model's behavior, explaining the empirical improvements over standard GRPO. Due to space limitations, all formal theorems and proofs are deferred to Section F.

4. Experimental Results

We evaluate the effectiveness of CoDistill-GRPO on both Qwen and Llama model families across four mathematical benchmarks: Minerva, MATH500, AMC2024, and OlympiadBench. For the Qwen models, we use Qwen2.5-Math-1.5B as the small model and Qwen2.5-Math-7B as the large model. These models were supervised fine-tuned on a mathematical dataset, making them well-suited for mathematical reasoning tasks and consistent with prior work. For the Llama family, we use Llama3.2-1B-Instruct and Llama3.1-8B-Instruct. All models are trained on the MATH dataset (Hendrycks et al., 2021), which consists of 12,500 samples, for a total of 8 epochs for Qwen models and 2 epochs for Llama models. We consider two verifiable reward functions—accuracy and format rewards—as done by (Shao et al., 2024). A detailed description of training configurations is deferred to Section B.

We compare CoDistill-GRPO, using different values of α

Algorithm	Minerva	MATH500	AMC	OlympiadBench	Average
Qwen2.5-Math-1.5B	20.37 ± 0.02	54.92 ± 0.01	32.00 ± 0.05	23.14 ± 0.01	32.61
+GRPO	25.88 ± 0.01	72.12 ± 0.01	55.50 ± 0.07	35.05 ± 0.00	47.14 (+14.53)
+KDRL w/ 7B	27.13 ± 0.02	74.24 ± 0.01	54.00 ± 0.05	$\mathbf{37.78} \pm 0.02$	48.29 (+15.68)
+CoDistill-GRPO ($\alpha = 1, T = 0$)	$\underline{29.12} \pm 0.01$	$\mathbf{75.04} \pm 0.01$	54.50 ± 0.04	$\underline{36.39} \pm 0.01$	48.76 (+16.15)
+CoDistill-GRPO ($\alpha = 2, T = 0$)	$\mathbf{31.99} \pm 0.02$	74.28 ± 0.01	$\underline{55.50} \pm 0.06$	35.14 ± 0.01	$\underline{49.23}$ (+16.62)
+CoDistill-GRPO ($\alpha = 1, T = 50$)	28.30 ± 0.01	$\underline{74.52} \pm 0.01$	$\mathbf{60.50} \pm 0.03$	36.12 ± 0.02	**49.86** (+17.25)

Table 1. Results for the small model Qwen2.5-Math-1.5B using CoDistill-GRPO compared to GRPO and KDRL. While all methods improve upon the base model, CoDistill-GRPO yields the largest overall improvement when averaged across datasets. Best results are shown in **bold**, and second-best results are underlined.

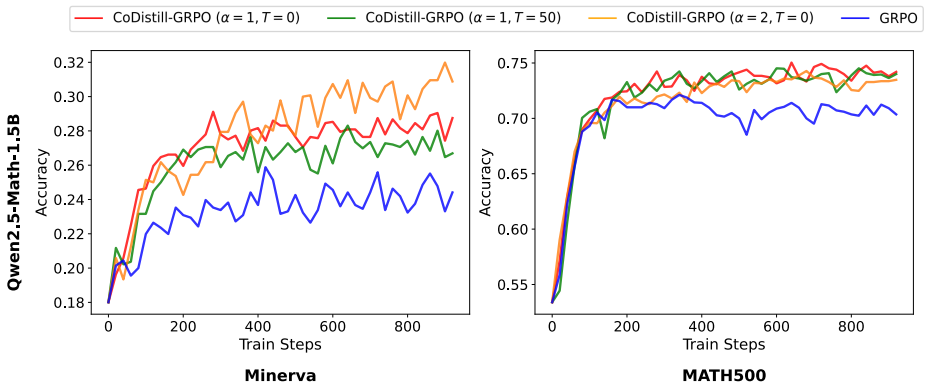

Figure 3. Test accuracy of the Qwen2.5-Math-1.5B model across training iterations on the Minerva and MATH500 datasets. Compared to GRPO, CoDistill-GRPO learns at a faster rate and can achieve a higher final test accuracy.

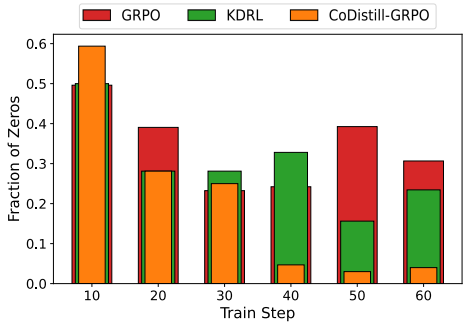

Figure 4. Fraction of zero accuracy rewards across training iterations on the MATH dataset (Hendrycks et al., 2021) for Qwen2.5-Math-1.5B. Compared to GRPO and KDRL, CoDistill-GRPO has a lesser fraction of zero accuracy rewards in the earlier iterations.

and initial hint lengths T, against two baselines: GRPO and KDRL (Xu et al., 2025a). For KDRL, we first run GRPO on the large model, select several well-performing checkpoints, and then train the small model using the effective on-policy KD reward. We report the results with the best performing checkpoint. Note that although KDRL freezes the large model during training, it still requires running GRPO beforehand, making it the most time-consuming method. We also fix $\alpha = 1$ for KDRL, as we empirically found that using a smaller fixed value performed better than using an annealed schedule. For CoDistill-GRPO, we provide hints, i.e., $T > 0$, only during the first epoch and set $T = 0$ for all subsequent iterations. For rollout generation, we use down-sampling by generating $M = 14$ rollouts and updating with a subset of $G = 8$ rollouts. This was also done for KDRL to provide a fair comparison. For GRPO, we generate and update using only $G = 8$ rollouts.

4.1. Results on Qwen Models

In Table 1, we present quantitative results on mathematical benchmarks for the Qwen2.5-Math-1.5B model. We report mean accuracy and standard deviation across five random seeds. While all methods improve upon the base model, CoDistill-GRPO achieves a notable improvement over GRPO with the small model alone, with an increase of approximately 2.7 percentage points on average accuracy. The small model trained with CoDistill-GRPO also reaches a remarkable 32% accuracy on the Minerva dataset as opposed to 26% with GRPO. On the other hand, while KDRL improves upon standard GRPO, its gains are marginal, yielding an average performance increase of 1.15 percentage point. We hypothesize that CoDistill-GRPO outperforms KDRL because jointly updating both models mitigates the previously discussed discrepancy between the large teacher and small student models. We also remark that while CoDistill-GRPO with large model hints (i.e., $T > 0$) achieves the best average performance increase, there are datasets where it does not improve upon the $T = 0$ base-

Algorithm	Minerva	MATH500	AMC	OlympiadBench	Average
Qwen2.5-Math-7B	24.04 ± 0.01	66.12 ± 0.01	47.00 ± 0.04	34.19 ± 0.01	42.84
+GRPO	$\mathbf{38.75} \pm 0.01$	80.12 ± 0.01	$\mathbf{70.00} \pm 0.01$	$\underline{42.73} \pm 0.01$	57.90 (+15.06)
+CoDistill-GRPO ($\alpha = 1, T = 0$)	34.56 ± 0.01	77.44 ± 0.01	64.00 ± 0.07	38.90 ± 0.01	53.73 (+10.89)
+CoDistill-GRPO ($\alpha = 1, T = 0$) w/ CT	$\underline{38.16} \pm 0.01$	79.04 ± 0.00	$\underline{69.50} \pm 0.03$	42.40 ± 0.01	57.27 (+14.43)
+CoDistill-GRPO ($\alpha = 1, T = 50$)	35.51 ± 0.01	77.04 ± 0.01	62.00 ± 0.02	40.41 ± 0.01	53.74 (+10.90)
+CoDistill-GRPO ($\alpha = 1, T = 50$) w/ CT	37.28 ± 0.02	$\mathbf{81.12} \pm 0.01$	63.50 ± 0.05	$\mathbf{42.96} \pm 0.01$	56.22 (+13.38)
+CoDistill-GRPO ($\alpha = 2, T = 0$)	35.15 ± 0.01	76.52 ± 0.01	62.00 ± 0.07	38.84 ± 0.01	53.13 (+10.29)
+CoDistill-GRPO ($\alpha = 2, T = 0$) w/ CT	36.00 ± 0.01	$\underline{80.20} \pm 0.00$	63.00 ± 0.02	41.01 ± 0.01	55.05 (+12.21)

Table 2. Results for the large model Qwen2.5-Math-7B using CoDistill-GRPO compared to baselines on Minerva, MATH500, AMC2024, and OlympiadBench. While all methods improve upon the base model, GRPO yields the largest overall improvement. However, with CT, CoDistill-GRPO can closely match the performance of GRPO while still reducing total training time. Best results are shown in **bold**, and second-best results are underlined.

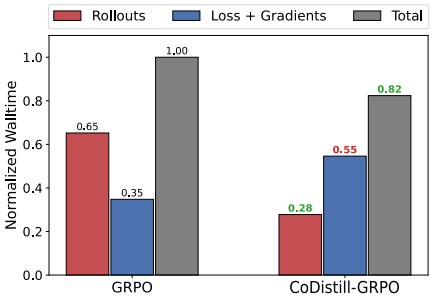

Figure 5. Demonstration of the potential time savings using CoDistill-GRPO over GRPO to train a Qwen2.5-Math-7B model on a single 80GB H100 GPU. Even though CoDistill-GRPO updates two models on every iteration (the small and the large), it achieves a reduction in training time due to rollout generation.

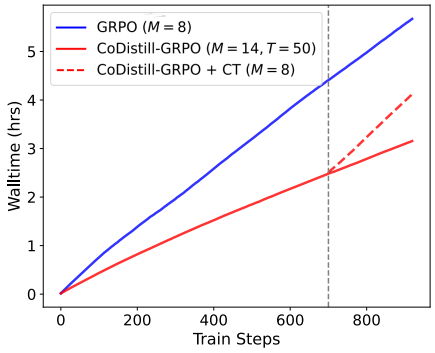

Figure 6. Cumulative walltime in hours for rollout generation. Even with CT, CoDistill-GRPO has a reduction in training time on Qwen2.5-Math-7B.

line. This implies that hinting is not universally beneficial, as the small model is often unable to complete the initial rollouts generated by the large model correctly. In Figure 3, we show the evolution of test accuracy on Minerva and MATH500 throughout training, and defer the AMC and OlympiadBench plots to Section D. Referring back to Figure 1, we show in Figure 4 that, compared to GRPO and KDRL, CoDistill-GRPO yields a smaller fraction of zero-accuracy rewards. We attribute this reduction to the gains observed in the quantitative results. Finally, in Section C, we provide additional ablation studies, including several settings in which CoDistill-GRPO was unstable.

4.2. How Much Does CoDistill-GRPO Improve the Large Model?

Since CoDistill-GRPO updates both models, an important question is how much improvement we can expect in the large model relative to standard GRPO. We compare standard GRPO on the Qwen2.5-Math-7B large model against both CoDistill-GRPO and CoDistill-GRPO with continued training (CT). CT serves as a post-processing step: we take a large model checkpoint from CoDistill-GRPO and run a few additional iterations of standard GRPO. This allows the model to use its own rollouts to boost performance, rather than relying only on the small model's rollouts.

In Table 2, we present the quantitative results, where we observe that CoDistill-GRPO with CT outperforms standard GRPO on the MATH500 and OlympiadBench. Of course, CoDistill-GRPO without CT does not outperform standard GRPO, as small model rollouts alone are not sufficient for substantial improvements. In Figure 10, we show the evolution of test accuracy during training, indicating where CT is introduced and demonstrating how much it improves the large model's performance. In Figure 5, we illustrate the time savings of CoDistill-GRPO compared to standard GRPO. By measuring the execution time of a single training iteration, we demonstrate an 18% reduction per step for CoDistill-GRPO when excluding CT and downsampling. In Figure 6, we present the total wall-clock time required strictly for rollout generation over a full training run. Here, standard GRPO generates $G = 8$ rollouts, whereas CoDistill-GRPO generates $M = 14$ rollouts prior to downsampling. We show that even when factoring in the overhead of CT, hinting (since $T = 50$), and downsampling, CoDistill-GRPO provides overall training-time savings during rollout generation.

References

Agarwal, R., Vieillard, N., Zhou, Y., Stanczyk, P., Garea, S. R., Geist, M., and Bachem, O. On-policy distillation of language models: Learning from self-generated mistakes. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=3zKtaqxLhW.

Ahmadian, A., Cremer, C., Gallé, M., Fadaee, M., Kreutzer, J., Pietquin, O., Üstün, A., and Hooker, S. Back to basics: Revisiting reinforce style optimization for learning from human feedback in llms. *arXiv preprint arXiv:2402.14740*, 2024. URL https://arxiv.org/abs/2402.14740.

Allen-Zhu, Z. and Li, Y. Towards understanding ensemble, knowledge distillation and self-distillation in deep learning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=Uuf2q9TfXGA.

Cho, J. H. and Hariharan, B. On the efficacy of knowledge distillation. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4793–4801, 2019. URL https://api.semanticscholar.org/CorpusID:203642130.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. URL https://arxiv.org/abs/2110.14168.

DeepSeek-AI, Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., Zhang, X., Yu, X., Wu, Y., Wu, Z. F., Gou, Z., Shao, Z., Li, Z., Gao, Z., Liu, A., Xue, B., Wang, B., Wu, B., Feng, B., Lu, C., Zhao, C., Deng, C., Zhang, C., Ruan, C., Dai, D., Chen, D., Ji, D., Li, E., Lin, F., Dai, F., Luo, F., Hao, G., Chen, G., Li, G., Zhang, H., Bao, H., Xu, H., Wang, H., Ding, H., Xin, H., Gao, H., Qu, H., Li, H., Guo, J., Li, J., Wang, J., Chen, J., Yuan, J., Qiu, J., Li, J., Cai, J. L., Ni, J., Liang, J., Chen, J., Dong, K., Hu, K., Gao, K., Guan, K., Huang, K., Yu, K., Wang, L., Zhang, L., Zhao, L., Wang, L., Zhang, L., Xu, L., Xia, L., Zhang, M., Zhang, M., Tang, M., Li, M., Wang, M., Li, M., Tian, N., Huang, P., Zhang, P., Wang, Q., Chen, Q., Du, Q., Ge, R., Zhang, R., Pan, R., Wang, R., Chen, R. J., Jin, R. L., Chen, R., Lu, S., Zhou, S., Chen, S., Ye, S., Wang, S., Yu, S., Zhou, S., Pan, S., Li, S. S., Zhou, S., Wu, S., Ye, S., Yun, T., Pei, T., Sun, T., Wang, T., Zeng, W., Zhao, W., Liu, W., Liang, W., Gao, W., Yu, W., Zhang, W., Xiao, W. L., An, W., Liu, X., Wang, X., Chen, X., Nie, X., Cheng, X., Liu, X., Xie, X., Liu, X., Yang, X., Li, X., Su, X., Lin, X., Li, X. Q., Jin, X., Shen, X., Chen, X., Sun, X., Wang, X., Song, X., Zhou, X., Wang, X., Shan, X., Li, Y. K., Wang, Y. Q., Wei, Y. X., Zhang, Y., Xu, Y., Li, Y., Zhao, Y., Sun, Y., Wang, Y., Yu, Y., Zhang, Y., Shi, Y., Xiong, Y., He, Y., Piao, Y., Wang, Y., Tan, Y., Ma, Y., Liu, Y., Guo, Y., Ou, Y., Wang, Y., Gong, Y., Zou, Y., He, Y., Xiong, Y., Luo, Y., You, Y., Liu, Y., Zhou, Y., Zhu, Y. X., Xu, Y., Huang, Y., Li, Y., Zheng, Y., Zhu, Y., Ma, Y., Tang, Y., Zha, Y., Yan, Y., Ren, Z. Z., Ren, Z., Sha, Z., Fu, Z., Xu, Z., Xie, Z., Zhang, Z., Hao, Z., Ma, Z., Yan, Z., Wu, Z., Gu, Z., Zhu, Z., Liu, Z., Li, Z., Xie, Z., Song, Z., Pan, Z., Huang, Z., Xu, Z., Zhang, Z., and Zhang, Z. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025. URL https://arxiv.org/abs/2501.12948.

Gu, Y., Dong, L., Wei, F., and Huang, M. MiniLLM: Knowledge distillation of large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=5h0qf7IBZZ.

Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., and Steinhardt, J. Measuring mathematical problem solving with the MATH dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. URL https://openreview.net/forum?id=7Bywt2mQsCe.

Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. URL https://arxiv.org/abs/1503.02531.

Hu, J., Zhang, Y., Han, Q., Jiang, D., Zhang, X., and Shum, H.-Y. Open-reasoner-zero: An open source approach to scaling up reinforcement learning on the base model. *arXiv preprint arXiv:2503.24290*, 2025. URL https://arxiv.org/abs/2503.24290.

Huang, T., You, S., Wang, F., Qian, C., and Xu, C. Knowledge distillation from a stronger teacher. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=157Usp_kbi.

Kaplun, G., eran malach, Nakkiran, P., and Shalev-Shwartz, S. Knowledge distillation: Bad models can be good role models. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=0ISChqjlrq.

Kim, Y. and Rush, A. M. Sequence-level knowledge distillation. In Su, J., Duh, K., and Carreras, X. (eds.),

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pp. 1317–1327, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1139. URL https://aclanthology.org/D16-1139/.

Lambert, N., Morrison, J., Pyatkin, V., Huang, S., Ivison, H., Brahman, F., Miranda, L. J. V., Liu, A., Dziri, N., Lyu, X., Gu, Y., Malik, S., Graf, V., Hwang, J. D., Yang, J., Bras, R. L., Tafjord, O., Wilhelm, C., Soldaini, L., Smith, N. A., Wang, Y., Dasigi, P., and Hajishirzi, H. Tulu 3: Pushing frontiers in open language model post-training. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=i1uGbfHHpH.

Liu, B., Wang, A., Min, Z., Yao, L., Zhang, H., Liu, Y., Han, X., Li, P., Zeng, A., and Su, J. Spec-rl: Accelerating on-policy reinforcement learning with speculative rollouts. *arXiv preprint arXiv:2509.23232*, 2026. URL https://arxiv.org/abs/2509.23232.

Liu, Z., Chen, C., Li, W., Qi, P., Pang, T., Du, C., Lee, W. S., and Lin, M. Understanding r1-zero-like training: A critical perspective. In *Conference on Language Modeling (COLM)*, 2025.

Lu, K. and Lab, T. M. On-policy distillation. *Thinking Machines Lab: Connectionism*, 2025. doi: 10.64434/tml.20251026. https://thinkingmachines.ai/blog/on-policy-distillation.

Menon, A. K., Rawat, A. S., Reddi, S., Kim, S., and Kumar, S. A statistical perspective on distillation. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 7632–7642. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/menon21a.html.

MiniMax, :, Chen, A., Li, A., Gong, B., Jiang, B., Fei, B., Yang, B., Shan, B., Yu, C., Wang, C., Zhu, C., Xiao, C., Du, C., Zhang, C., Qiao, C., Zhang, C., Du, C., Guo, C., Chen, D., Ding, D., Sun, D., Li, D., Jiao, E., Zhou, H., Zhang, H., Ding, H., Sun, H., Feng, H., Cai, H., Zhu, H., Sun, J., Zhuang, J., Cai, J., Song, J., Zhu, J., Li, J., Tian, J., Liu, J., Xu, J., Yan, J., Liu, J., He, J., Feng, K., Yang, K., Xiao, K., Han, L., Wang, L., Yu, L., Feng, L., Li, L., Zheng, L., Du, L., Yang, L., Zeng, L., Yu, M., Tao, M., Chi, M., Zhang, M., Lin, M., Hu, N., Di, N., Gao, P., Li, P., Zhao, P., Ren, Q., Xu, Q., Li, Q., Wang, Q., Tian, R., Leng, R., Chen, S., Chen, S., Shi, S., Weng, S., Guan, S., Yu, S., Li, S., Zhu, S., Li, T., Cai, T., Liang, T., Cheng, W., Kong, W., Li, W., Chen, X., Song, X., Luo, X., Su, X., Li, X., Han, X., Hou, X., Lu, X., Zou, X., Shen, X., Gong, Y., Ma, Y., Wang, Y., Shi, Y., Zhong, Y., Duan, Y., Fu, Y., Hu, Y., Gao, Y., Fan, Y., Yang, Y., Li, Y., Hu, Y., Huang, Y., Li, Y., Xu, Y., Mao, Y., Shi, Y., Wenren, Y., Li, Z., Li, Z., Tian, Z., Zhu, Z., Fan, Z., Wu, Z., Xu, Z., Yu, Z., Lyu, Z., Jiang, Z., Gao, Z., Wu, Z., Song, Z., and Sun, Z. Minimax-m1: Scaling test-time compute efficiently with lightning attention. *arXiv preprint arXiv:2506.13585*, 2025. URL https://arxiv.org/abs/2506.13585.

Mirzadeh, S. I., Farajtabar, M., Li, A., Levine, N., Matsukawa, A., and Ghasemzadeh, H. Improved knowledge distillation via teacher assistant. In *AAAI Conference on Artificial Intelligence*, 2019. URL https://api.semanticscholar.org/CorpusID:212908749.

Nagarajan, V., Menon, A. K., Bhojanapalli, S., Mobahi, H., and Kumar, S. On student-teacher deviations in distillation: does it pay to disobey? In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=7UdVPRmpif.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Gray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., and Lowe, R. Training language models to follow instructions with human feedback. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022a. URL https://openreview.net/forum?id=TG8KACxEON.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P. F., Leike, J., and Lowe, R. Training language models to follow instructions with human feedback. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 27730–27744. Curran Associates, Inc., 2022b. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/b1efde53be364a73914f58805a001731-Paper-Conference.pdf.

Phuong, M. and Lampert, C. Towards understanding knowledge distillation. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5142–5151. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/phuong19a.html.

Rawat, A. S., Sadhanala, V., Rostamizadeh, A., Chakrabarti, A., Jitkrittum, W., Feinberg, V., Kim, S., Harutyunyan,

H., Saunshi, N., Nado, Z., Shivanna, R., Reddi, S. J., Menon, A. K., Anil, R., and Kumar, S. A little help goes a long way: Efficient llm training by leveraging small lms. *arXiv preprint arXiv:2410.18779*, 2024. URL https://arxiv.org/abs/2410.18779.

Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y. K., Wu, Y., and Guo, D. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024. URL https://arxiv.org/abs/2402.03300.

Shrivastava, V., Awadallah, A., Balachandran, V., Garg, S., Behl, H., and Papailiopoulos, D. Sample more to think less: Group filtered policy optimization for concise reasoning. *arXiv preprint arXiv:2508.09726*, 2025. URL https://arxiv.org/abs/2508.09726.

Snell, C., Klein, D., and Zhong, R. Learning by distilling context. *arXiv preprint arXiv:2209.15189*, 2022. URL https://arxiv.org/abs/2209.15189.

Wen, X., Liu, Z., Zheng, S., Ye, S., Wu, Z., Wang, Y., Xu, Z., Liang, X., Li, J., Miao, Z., Bian, J., and Yang, M. Reinforcement learning with verifiable rewards implicitly incentivizes correct reasoning in base LLMs. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=jGbRWwIidy.

Xiao, C., Zhang, M., and Cao, Y. Bnpo: Beta normalization policy optimization. *arXiv preprint arXiv:2506.02864*, 2025. URL https://arxiv.org/abs/2506.02864.

Xu, H., Zhu, Q., Deng, H., Li, J., Hou, L., Wang, Y., Shang, L., Xu, R., and Mi, F. Kdrl: Post-training reasoning llms via unified knowledge distillation and reinforcement learning. *arXiv preprint arXiv:2506.02208*, 2025a. URL https://arxiv.org/abs/2506.02208.

Xu, X., Li, M., Tao, C., Shen, T., Cheng, R., Li, J., Xu, C., Tao, D., and Zhou, T. A survey on knowledge distillation of large language models. *arXiv preprint arXiv:2402.13116*, 2024. URL https://arxiv.org/abs/2402.13116.

Xu, Y. E., Savani, Y., Fang, F., and Kolter, J. Z. Not all rollouts are useful: Down-sampling rollouts in llm reinforcement learning. *arXiv preprint arXiv:2504.13818*, 2025b. URL https://arxiv.org/abs/2504.13818.

Yu, Q., Zhang, Z., Zhu, R., Yuan, Y., Zuo, X., Yue, Y., Dai, W., Fan, T., Liu, G., Liu, L., Liu, X., Lin, H., Lin, Z., Ma, B., Sheng, G., Tong, Y., Zhang, C., Zhang, M., Zhang, W., Zhu, H., Zhu, J., Chen, J., Chen, J., Wang, C., Yu, H., Song, Y., Wei, X., Zhou, H., Liu, J., Ma, W.-Y., Zhang, Y.-Q., Yan, L., Qiao, M., Wu, Y., and Wang, M. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025. URL https://arxiv.org/abs/2503.14476.

Yuan, W., Pang, R. Y., Cho, K., Li, X., Sukhbaatar, S., Xu, J., and Weston, J. Self-rewarding language models. *arXiv preprint arXiv:2401.10020*, 2025. URL https://arxiv.org/abs/2401.10020.

Yue, Y., Chen, Z., Lu, R., Zhao, A., Wang, Z., Yue, Y., Song, S., and Huang, G. Does reinforcement learning really incentivize reasoning capacity in LLMs beyond the base model? In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=4OsgYD7em5.

Zhang, K., Hong, Y., Bao, J., Jiang, H., yang song, Dingqian, H., and Xiong, H. GVPO: Group variance policy optimization for large language model post-training. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025a. URL https://openreview.net/forum?id=cCYUFaR6En.

Zhang, X., Huang, Z., Li, Y., Ni, C., Chen, J., and Oymak, S. BREAD: Branched rollouts from expert anchors bridge SFT & RL for reasoning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025b. URL https://openreview.net/forum?id=NUDaln2vCe.

Zhang, X., Wu, S., Zhu, Y., Tan, H., Yu, S., He, Z., and Jia, J. Scaf-GRPO: Scaffolded group relative policy optimization for enhancing LLM reasoning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=bOwVr0yr7r.

Zhao, Y., Liu, Y., Liu, J., Chen, J., Wu, X., Hao, Y., Lv, T., Huang, S., Cui, L., Ye, Q., Wan, F., and Wei, F. Geometric-mean policy optimization. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=nCEs0tSwc2.

Zheng, C., Liu, S., Li, M., Chen, X.-H., Yu, B., Gao, C., Dang, K., Liu, Y., Men, R., Yang, A., Zhou, J., and Lin, J. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*, 2025a. URL https://arxiv.org/abs/2507.18071.

Zheng, H., Zhou, Y., Bartoldson, B. R., Kailkhura, B., Lai, F., Zhao, J., and Chen, B. Act only when it pays: Efficient reinforcement learning for LLM reasoning via selective rollouts. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*,

2025b. URL https://openreview.net/forum?id=x5lITYXmW2.

Appendix

A. Related Work

Here, we survey a few works most closely related to our work.

Improving GRPO. There is a growing body of work aimed at improving GRPO (Xiao et al., 2025; Liu et al., 2025; Zheng et al., 2025a; Yu et al., 2025; Shrivastava et al., 2025; Xu et al., 2025b; Zhang et al., 2025a; MiniMax et al., 2025; Zhao et al., 2026). CoDistill-GRPO can be combined with these methods, as our contribution primarily provides an algorithmic framework for co-distillation rather than a direct modification of GRPO. On the other hand, there are two more closely related works on improving GRPO for small models: BREAD (Zhang et al., 2025b) and KDRL (Xu et al., 2025a). As previously stated, both algorithms start from the assumption that a larger, more powerful language model is available to assist a smaller model. In contrast to BREAD, CoDistill-GRPO starts with the large model generating a fixed number of initial tokens as hints for each rollout. This procedure is applied only during the early iterations to help align the two model distributions, enabling rollouts from the small model to be used to update the large model. KDRL shares the closest structural similarity to our algorithm and serves as our main comparison. It uses a KD-based method to improve small model performance while also maximizing the GRPO objective. However, (Xu et al., 2025a) keeps the large model fixed: we show that jointly training models via CoDistill-GRPO yields better performance than merely applying GRPO with a KD-based reward from a static teacher model.

Knowledge Distillation. KD is an effective method for reducing the computational overhead of LLMs by distilling knowledge from large models into smaller ones (Hinton et al., 2015; Kim & Rush, 2016; Agarwal et al., 2024; Gu et al., 2024; Xu et al., 2024; Phuong & Lampert, 2019; Menon et al., 2021; Kaplun et al., 2022; Nagarajan et al., 2023; Rawat et al., 2024). This is typically achieved by minimizing the KL divergence between the probability distributions of the student and teacher models. Our work is inspired by the on-policy distillation framework for LLMs proposed by (Agarwal et al., 2024), and similarly by (Lu & Lab, 2025), who introduce an optimization objective that maximizes a reward augmented with an on-policy KL term for training a smaller student model. We remark that there also exists a line of work on "self-distillation" (Snell et al., 2022; Yuan et al., 2025; Allen-Zhu & Li, 2023), but our goal is to directly use two different models to improve the performance of the smaller model.

B. Training Details

Hyperparameter	Qwen Models	Llama Models
Learning Rate	5×10^{-6}	1×10^{-7}
Learning Rate Schedule	Cosine	Constant
Learning Rate Warmup Steps	5%	0%
Max Completion Length	3072	3072
Max Prompt Length	1024	1024
Batch Size Per GPU	64	64
Number of Rollouts	$M = 14, G = 8$	$M = 12, G = 8$
Total Epochs	8	2

Table 3. Hyperparameter configurations used to train both Qwen and Llama models. We use a constant learning rate with no warmup steps for Llama models, as this yielded the best performance. For CoDistill-GRPO, we generate an initial M traces, for which G are chosen using the downsampling method. For GRPO, we generate and update using G rollouts.

In this section, we present the omitted training details that support the experiments in Section 4. All experiments were conducted using 8 H100 GPUs unless otherwise specified. We report the hyperparameter settings in Table 3. The same

A conversation between User and Assistant. The user asks a question, and the Assistant solves it. The assistant first thinks about the reasoning process in the mind and then provides the user with the answer. The reasoning process and answer are enclosed within `<think></think>` and `<answer></answer>` tags, respectively, i.e., `<think> reasoning process here </think>` and `<answer> answer here </answer>`. User: {prompt}. Assistant:

Table 4. Chat template for DeepSeek-R1-Zero (DeepSeek-AI et al., 2025) that is used to train the Qwen models for both GRPO and CoDistill-GRPO. The prompt is replaced with the specific reasoning question during training.

Please reason step by step, and put your final answer within `\boxed{}`.
Question: {prompt}. Answer:

Table 5. Chat template used to train the Llama models for both GRPO and CoDistill-GRPO. The prompt is replaced with the specific reasoning question during training.

set of hyperparameters was used for both the small and large models, as well as for CoDistill-GRPO and GRPO, as this configuration yielded the best performance. For the number of rollouts, M denotes the number of rollouts initially generated for CoDistill-GRPO, while G denotes the number of rollouts selected after downsampling. For GRPO, we directly generate G rollouts and perform updates using these G samples. In experiments with Llama models, we found that using a constant learning rate achieved the best performance on the test datasets. We also observed that increasing the number of epochs was detrimental, as longer training led to instability.

Following DeepSeek-R1 (Shao et al., 2024; DeepSeek-AI et al., 2025), we add an additional format reward for Qwen models to encourage adherence to a specified output format. The format used for the Qwen models is shown in Table 4. The model receives a reward of 0.25 for each of the `</think>` and `</answer>` tokens, for a total possible format reward of 1.00. For accuracy rewards, each correct answer is also assigned a reward of 1.00. For the Llama models, we use the chat template shown in Table 5, as we observed that this template provides a substantial performance improvement compared to the template used for Qwen. However, we omit format rewards for Llama models and include only the accuracy reward to reflect this difference.

C. Ablation Studies

In this section, we present ablation studies investigating the algorithmic choices made in CoDistill-GRPO. All ablation studies were performed on the Qwen models. In Figure 7, we demonstrate how the KD reward and rejection-sampling components are crucial for simultaneously training both models. We observe that without these two components, the large model becomes unstable during training when evaluated on Minerva and MATH500. This shows that simply using small-model rollouts to update the large model is insufficient.

In Figure 8, we study the effect of using initial traces (or hints) from the large model throughout all training iterations, as opposed to only during the early iterations. On the left, we plot the training accuracy of the small model over time. When the training accuracy drops, the large model is updated using rollouts with low accuracy; consequently, the large model's performance degrades significantly and fails to recover. In contrast, the setting with $T = 0$ trains stably. On the right, we show that ignoring the initial traces provided by the large model in the importance-sampling update for the small-model objective also leads to unstable training. Finally, in Figure 9, we show that using the KL-divergence definition in Equation (4) results in training instability.

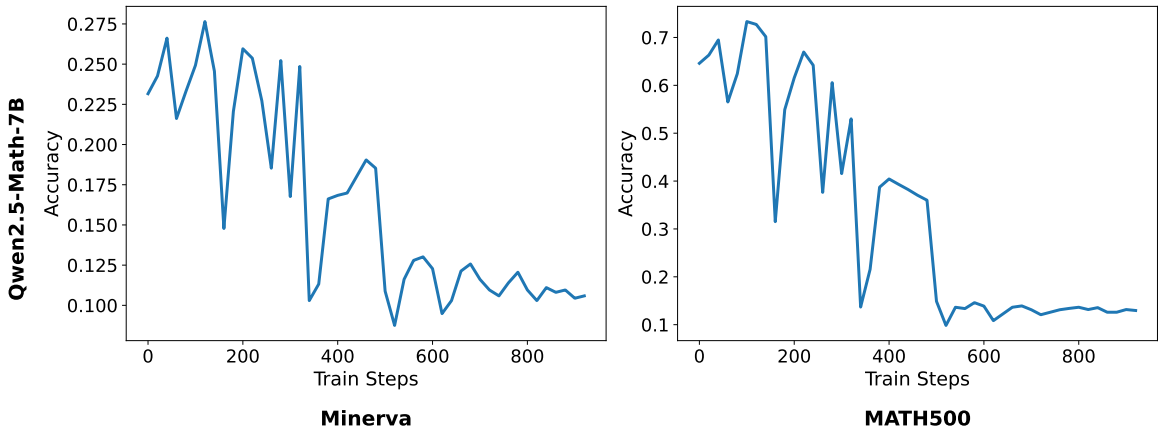

Figure 7. Using small model rollouts to update the policy model with no knowledge distillation or downsampling. These two components that make up CoDistill-GRPO are critical for making simultaneous training of models feasible.

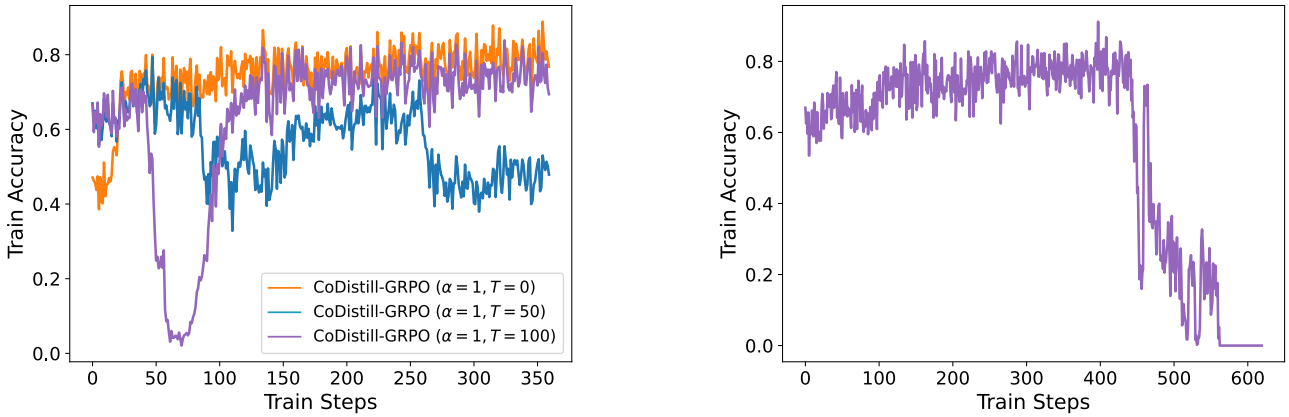

Figure 8. Left: Effects of using initial traces from the large model for all iterations of training for different values of T. Providing initial traces does not always help and makes training unstable. Right: Plot of the train accuracy of the small model for ignoring the initial traces in the importance sampling for CoDistill-GRPO as done in BREAD. Ignoring the initial traces makes training of CoDistill-GRPO unstable.

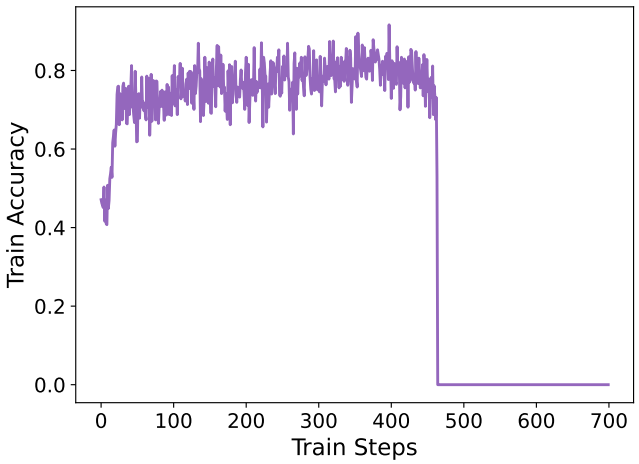

Figure 9. Plot of the train accuracy during training of the small model using the KL divergence as defined in Equation (4). Using Equation (4) is less robust as opposed to using the KL divergence in the reward of Equation (6).

D. Additional Figures

In this section, we present the test accuracy across training iterations that were deferred in Section 4.

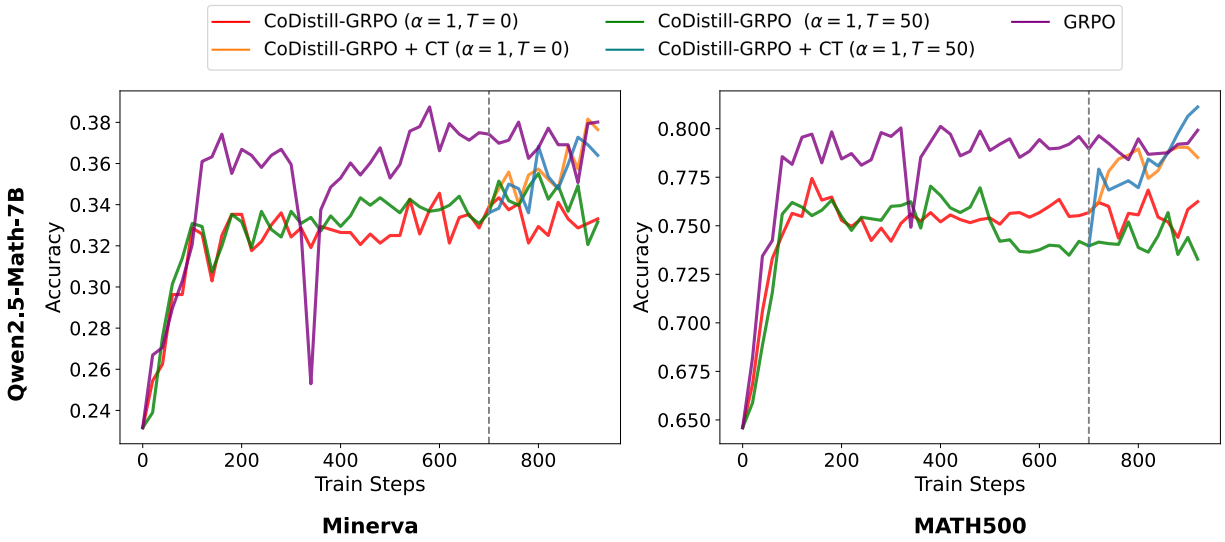

Figure 10. Test accuracy of the Qwen2.5-Math-7B model across training iterations on the Minerva and MATH500 datasets. While GRPO obtains the best performance averaged across all benchmark datasets, CoDistill-GRPO with continued training (CT) closely matches GRPO even while using rollouts from the smaller model.

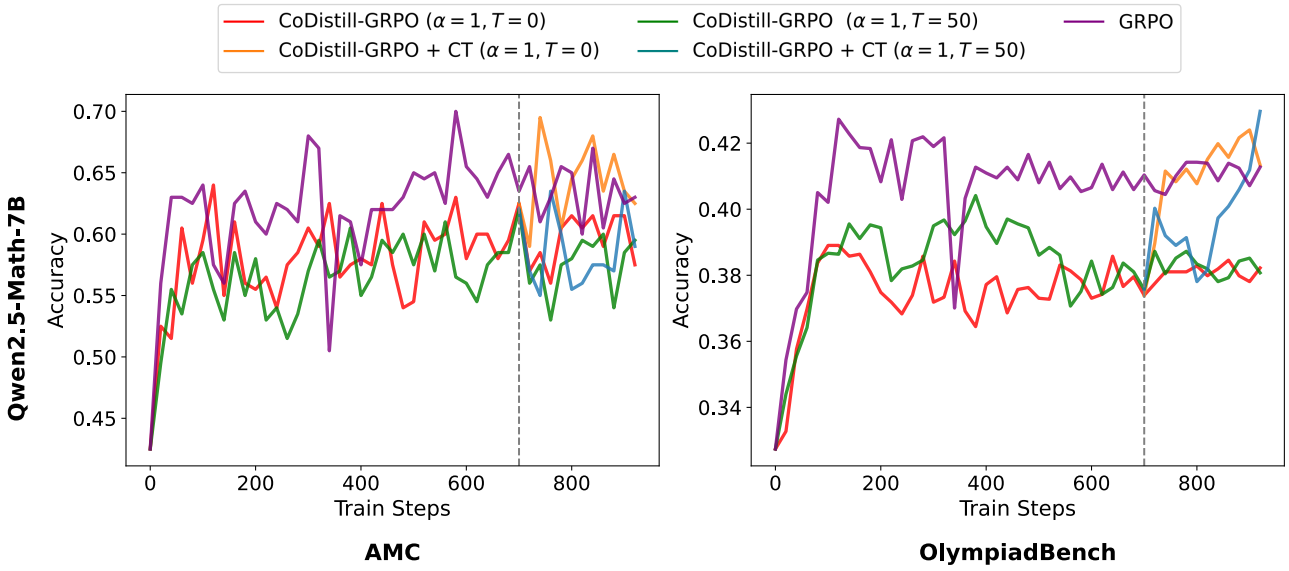

Figure 11. Test accuracy of the Qwen2.5-Math-7B model across training iterations on the AMC and OlympiadBench datasets.

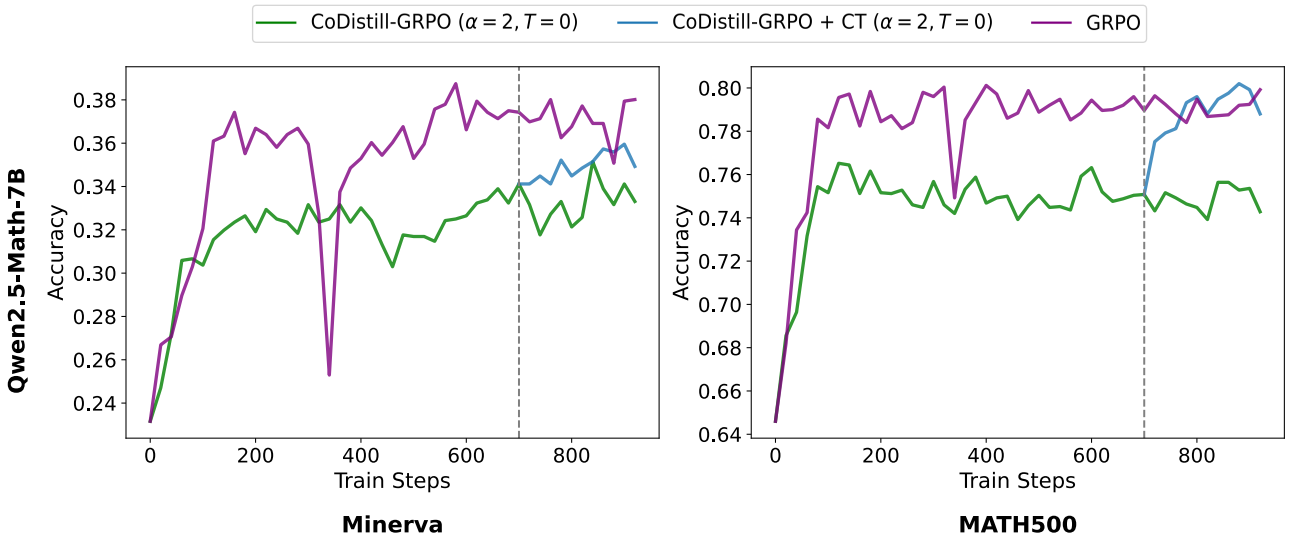

Figure 12. Test accuracy of the Qwen2.5-Math-7B model across training iterations on the Minerva and MATH500 datasets with $\alpha = 2, T = 0$.

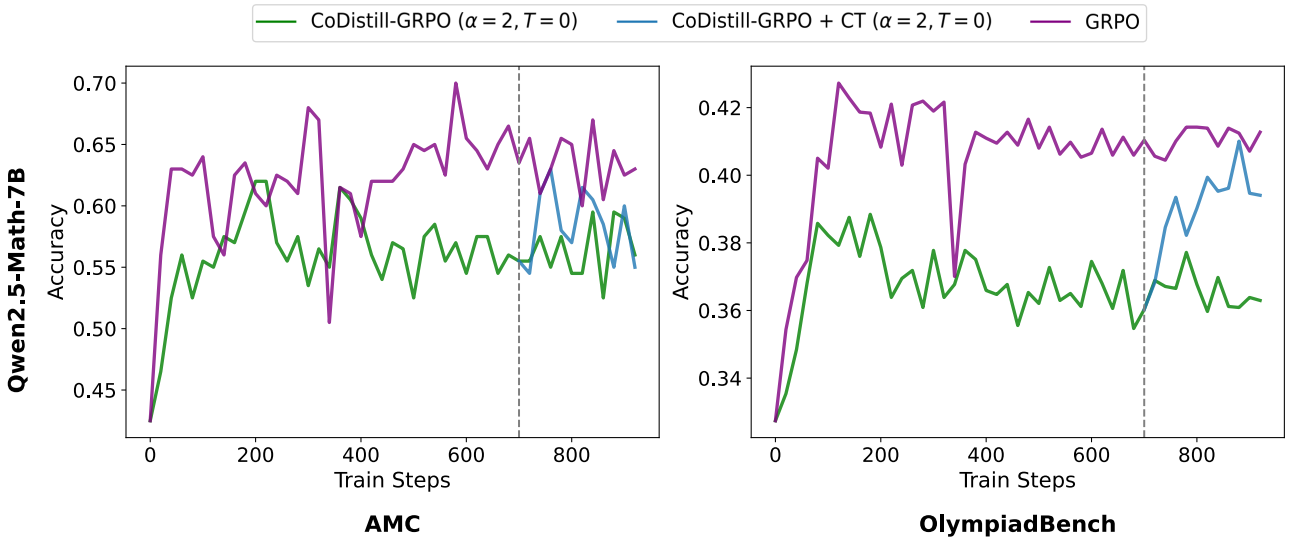

Figure 13. Test accuracy of the Qwen2.5-Math-7B model across training iterations on the AMC and OlympiadBench datasets with $\alpha = 2, T = 0$.

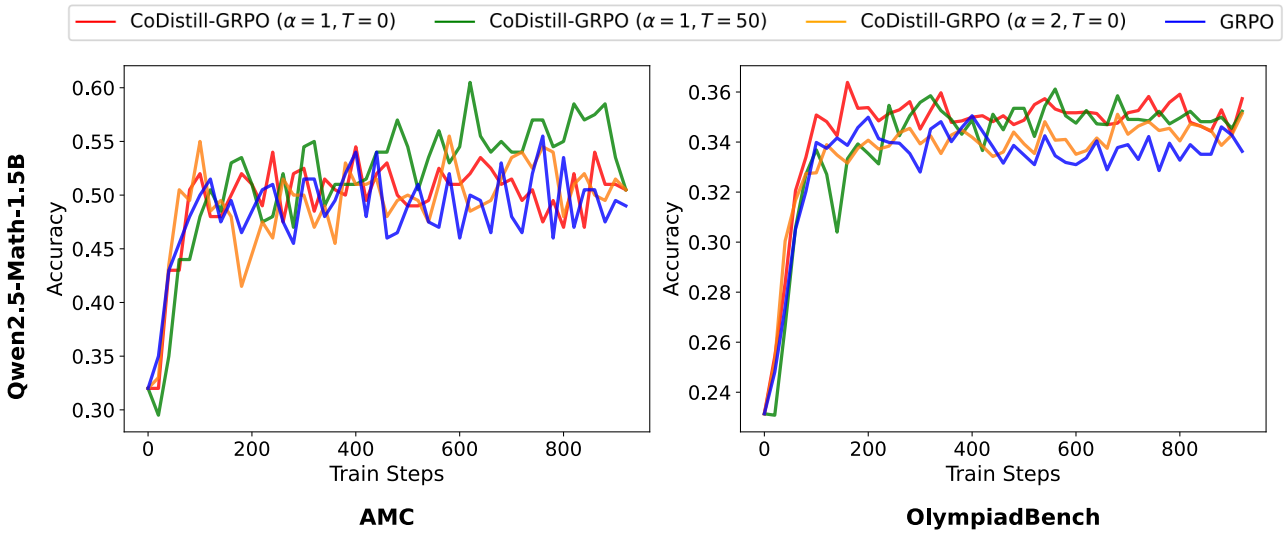

Figure 14. Test accuracy of the Qwen2.5-Math-1.5B model across training iterations on the AMC and OlympiadBench datasets.

E. Results on Llama Models

While Qwen models are generally the standard choice for mathematical benchmarks, we also demonstrate the effectiveness of CoDistill-GRPO on Llama models. In Table 6, we show that CoDistill-GRPO provides improvements over GRPO. Although both methods outperform the base model, the gains are considerably smaller than those observed with Qwen. This observation is consistent with findings in the literature (see the experiments on Llama models reported by Liu et al. (2025)). We also find that training Llama with GRPO is typically unstable, making benchmarking more challenging than with Qwen. In contrast, CoDistill-GRPO appeared substantially more stable and achieved consistent improvements over GRPO. We leave a more thorough investigation to future work.

Algorithm	Minerva	MATH500	AMC	OlympiadBench	Average
Llama3.2-1B-Instruct	6.465 ± 0.55	20.56 ± 2.20	5.500 ± 2.45	4.328 ± 0.22	9.213
+GRPO	5.664 ± 0.95	22.36 ± 2.47	6.500 ± 4.06	5.038 ± 0.57	9.891 (+0.678)
+CoDistill-GRPO ($\alpha = 1, T = 0$)	6.396 ± 1.50	22.30 ± 1.26	10.50 ± 1.87	5.086 ± 0.45	11.07 (+1.857)

Table 6. Results for the small model Llama3.2-1B-Instruct using CoDistill-GRPO compared to baselines on Minerva, MATH500, AMC2024, and OlympiadBench. CoDistill-GRPO improves upon GRPO when averaged across all benchmarking datasets.

F. Theoretical Results

In this section, we present the deferred theoretical results and their corresponding proofs. The following theorem demonstrates that the gradient estimate with respect to $\mathcal{L}_{\texttt{Small}}(\phi)$ remains unbiased.

Lemma F.1. *Let $\eta > 0$ and $G > 1$ denote the learning rate and group size, respectively. Suppose that for each prompt $q \sim P(Q)$, the outputs are sampled on-policy $\{o_i\}_{i=1}^{G} \sim \pi_\phi(\cdot \mid q)$ and π_ϕ is continuously differentiable. If the policy π_ϕ is updated using the effective learning rate $\eta_{\textit{eff}} \coloneqq \left(\frac{G}{G-1}\right) \cdot \eta$ with advantage estimates in Equation (6), then the gradient of Equation (5) is an unbiased estimate of the true policy gradient:*

$$\eta_{\textit{eff}} \cdot \mathbb{E}\left[\nabla_\phi \mathcal{L}_{\texttt{Small}}(\phi)\right] = \eta \cdot \nabla_\phi J(\phi), \tag{11}$$

where $J(\phi) \coloneqq \mathbb{E}_{q\sim P(Q), o_i \sim \pi_\phi(\cdot|q)}\left[\widetilde{r}_\phi(q, o_i)\right]$ is the true reward objective.

Proof. Recall that the reward $\widetilde{r}_i$ depends on the model parameters ϕ. Using the product rule, the gradient of the expected loss in Equation (5) with respect to ϕ yields

$$\nabla_\phi \mathcal{L}_{\texttt{Small}}(\phi)\,|_{\phi=\phi_{\text{old}}} = \frac{1}{G}\sum_{i=1}^{G}\sum_{t=1}^{N}\mathbb{E}\Big[\underbrace{\widetilde{A}_{i,t}\cdot\nabla_\phi \log \pi_\phi\left(o_{i,t} \mid q, o_{i,<t}\right)}_{\text{policy gradient}} + \underbrace{\nabla_\phi \widetilde{r}_i(\phi) - \nabla_\phi \text{mean}\left(\widetilde{\mathbf{r}}(\phi)\right)}_{\text{reward gradient from KD}}\Big],$$

where

$$\nabla_\phi \widetilde{r}_i(\phi) = -\alpha \cdot \frac{1}{N}\sum_{t=1}^{N}\nabla_\phi \log \pi_\phi\left(o_{i,t} \mid q, o_{i,<t}\right).$$

Notice that

$$\begin{aligned}
\mathbb{E}_{o_i\sim\pi_\phi(\cdot|q)}\left[\nabla_\phi \widetilde{r}_i(\phi)\right] &= -\alpha \cdot \frac{1}{N}\sum_{t=1}^{N}\mathbb{E}_{o_i\sim\pi_\phi(\cdot|q)}\left[\nabla_\phi \log \pi_\phi\left(o_{i,t} \mid q, o_{i,<t}\right)\right] && (12)\\
&= -\alpha \cdot \frac{1}{N}\sum_{t=1}^{N}\mathbb{E}_{o_{i,<t}}\left[\mathbb{E}_{o_{i,t}\sim\pi_\phi(\cdot|q,o_{i,<t})}\left[\nabla_\phi \log \pi_\phi\left(o_{i,t} \mid q, o_{i,<t}\right)\right]\right] && \text{(Tower Property)}\\
&= -\alpha \cdot \frac{1}{N}\sum_{t=1}^{N}\mathbb{E}_{o_{i,<t}}\left[\nabla_\phi \int \pi_\phi\left(o_{i,t} \mid q, o_{i,<t}\right)\right] && \text{(Leibniz Rule)}\\
&= -\alpha \cdot \frac{1}{N}\sum_{t=1}^{N}\nabla_\phi 1 && (13)\\
&= 0, && (14)
\end{aligned}$$

where the tower property was used for each token index t. Hence, the reward gradient from KD is zero in expectation, yielding the following remaining gradient:

$$\nabla_\phi \mathcal{L}_{\texttt{Small}}(\phi)\,|_{\phi=\phi_{\text{old}}} = \frac{1}{G}\sum_{i=1}^{G}\sum_{t=1}^{N}\mathbb{E}\Big[\widetilde{A}_{i,t}\cdot\nabla_\phi \log \pi_\phi\left(o_{i,t} \mid q, o_{i,<t}\right)\Big].$$

For simplicity in notation, let us denote

$$s_i \coloneqq \sum_{t=1}^{N}\nabla_\phi \log \pi_\phi\left(o_{i,t} \mid q, o_{i,<t}\right).$$

Then, using the fact that $\widetilde{A}_{i,t} = \widetilde{A}_i = \widetilde{r}_i - \text{mean}(\widetilde{\mathbf{r}})$, notice that

$$\begin{aligned}
\frac{1}{G}\sum_{i=1}^{G}\widetilde{A}_i \cdot s_i &= \frac{1}{G}\sum_{i=1}^{G}\widetilde{r}_i \cdot s_i - \frac{1}{G}\text{mean}(\widetilde{\mathbf{r}})\sum_{i=1}^{G}s_i\\
&= \frac{1}{G}\sum_{i=1}^{G}\widetilde{r}_i \cdot s_i - \frac{1}{G^2}\left(\sum_{j=1}^{G}\widetilde{r}_j\right)\left(\sum_{i=1}^{G}s_i\right).
\end{aligned}$$

By taking the conditional expectation on q, we obtain

$$\mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G}\widetilde{r}_i \cdot s_i \,|\, q\right] - \mathbb{E}\left[\frac{1}{G^2}\left(\sum_{j=1}^{G}\widetilde{r}_j\right)\left(\sum_{i=1}^{G}s_i\right) \,|\, q\right] = \mathbb{E}[\widetilde{r}_i s_i \,|\, q] - \frac{1}{G^2}\sum_{i}^{G}\mathbb{E}[\widetilde{r}_i s_i \,|\, q] - \frac{1}{G^2}\underbrace{\sum_{i\neq j}\mathbb{E}[\widetilde{r}_j s_i \,|\, q]}_{=0 \text{ (by independence)}}$$

$$= \left(\frac{G-1}{G}\right)\mathbb{E}\left[\widetilde{r}_i s_i \,|\, q\right]$$

Taking the average over q, we have

$$\nabla_\phi \mathcal{L}_{\texttt{Small}}(\phi) = \left(\frac{G-1}{G}\right)\cdot \mathbb{E}\left[\widetilde{r}_i \cdot \sum_{t=1}^{N}\nabla_\phi \log \pi_\phi\left(o_{i,t} \,|\, q, o_{i,<t}\right)\right]. \tag{15}$$

Now we derive the true policy gradient by defining the reward objective:

$$J(\phi) \coloneqq \mathbb{E}_{q\sim P(Q), o_i\sim\pi_\phi(\cdot|q)}\left[\widetilde{r}_\phi(q, o_i)\right],$$

which yields

$$\nabla_\phi J(\phi) = \mathbb{E}\left[\widetilde{r}_\phi(q,o_i)\cdot\nabla_\phi \log \pi_\phi\left(o_i \,|\, q\right)\right] + \underbrace{\mathbb{E}\left[\nabla_\phi \widetilde{r}_\phi(q,o_i)\right]}_{=0 \text{ (derived in Equation (12))}}$$

$$= \mathbb{E}\left[\widetilde{r}_\phi(q,o_i)\cdot\sum_{t=1}^{N}\nabla_\phi \log \pi_\phi\left(o_{i,t} \,|\, q, o_{i,<t}\right)\right].$$

Then by multiplying the learning rate η on both sides for the update rule:

$$\eta\cdot\nabla_\phi J(\phi) = \eta\cdot\mathbb{E}\left[\widetilde{r}_i\cdot\sum_{t=1}^{N}\nabla_\phi \log \pi_\phi\left(o_{i,t} \,|\, q, o_{i,<t}\right)\right]$$

$$= \underbrace{\eta\left(\frac{G}{G-1}\right)}_{=\eta_{\text{eff}}}\nabla_\phi \mathcal{L}_{\texttt{Small}}(\phi)$$

$$= \eta_{\text{eff}}\cdot\nabla_\phi \mathcal{L}_{\texttt{Small}}(\phi),$$

which completes the proof. We remark that the adjustment in learning rate also yields a gradient equivalent to the RLOO (Ahmadian et al., 2024) gradient, which suggests an alternative direction for the proof. □

Note that Theorem F.1 makes two assumptions: the outputs are sampled on-policy, meaning that importance reweighting is not performed; and only G outputs are sampled, meaning that the downsampling procedure is skipped. These assumptions simplify the analysis, allowing us to demonstrate that the algorithm's properties hold based solely on its core components. Then, using Theorem F.1, we prove that the gradient of $\mathcal{L}_{\texttt{Small}}(\phi)$ can be decomposed into two terms: a direction towards the original reward and a direction towards the large model. This shows that when $\alpha > 0$, the effective reward drives the small model towards the large model, explaining why the effective reward improves small model performance beyond standard GRPO.

Then, we prove that the gradient of $\mathcal{L}_{\texttt{Small}}(\phi)$ can be decomposed into two terms: a direction towards the original reward and a direction towards the large model.

Theorem F.2 (Gradient Decomposition)**.** *Let $\alpha > 0$ and $N > 0$ denote the KD regularization parameter and number of maximum completion tokens, respectively. Under the same conditions as Theorem F.1, for each prompt $q \sim P(Q)$, the expected gradient of $\mathcal{L}_{\texttt{Small}}(\phi)$ admits the following decomposition:*

$$\eta_{\text{eff}}\cdot\mathbb{E}\left[\nabla_\phi \mathcal{L}_{\texttt{Small}}(\phi) \,|\, q\right] = \eta\cdot\underbrace{\nabla_\phi \mathbb{E}_{o_i\sim\pi_\phi(\cdot|q)}\left[r(q,o_i)\right]}_{\textit{direction towards original reward}} - \eta\cdot\frac{\alpha}{N}\underbrace{\nabla_\phi \mathbb{D}_{KL}\left(\pi_\phi(\cdot|q)\|\pi_\theta(\cdot|q)\right)}_{\textit{direction towards large model}}.$$

Proof. Let us define

$$J_q(\phi) \coloneqq \mathbb{E}_{o_i \sim \pi_\phi(\cdot|q)} \left[\widetilde{r}_\phi(q, o_i) \right] \quad \text{such that} \quad J(\phi) = \mathbb{E}_{q \sim P(Q)} \left[J_q(\phi) \right].$$

Recall that by Theorem F.1, we have

$$\eta_{\text{eff}} \cdot \mathbb{E} \left[\nabla_\phi \mathcal{L}_{\texttt{Small}}(\phi) \,|\, q \right] = \eta \cdot \nabla_\phi J_q(\phi),$$

conditioned on q, before averaging over q. Hence, it suffices to decompose $J_q(\phi)$. Expanding the effective reward from Equation (6):

$$J_q(\phi) = \mathbb{E}_{o_i \sim \pi_\phi(\cdot|q)} \left[r(q, o_i) \right] + \alpha \cdot \mathbb{E}_{o_i \sim \pi_\phi(\cdot|q)} \left[\frac{1}{N} \sum_{t=1}^{N} \log \frac{\pi_\theta(o_{i,t}|q, o_{i,<t})}{\pi_\phi(o_{i,t}|q, o_{i,<t})} \right].$$

By the chain rule, we have

$$\sum_{t=1}^{N} \log \frac{\pi_\theta(o_{i,t}|q, o_{i,<t})}{\pi_\phi(o_{i,t}|q, o_{i,<t})} = \log \frac{\pi_\theta(o_i|q)}{\pi_\phi(o_i|q)},$$

and so the second term becomes

$$\frac{\alpha}{N} \cdot \mathbb{E}_{o_i \sim \pi_\phi(\cdot|q)} \left[\log \frac{\pi_\theta(o_i|q)}{\pi_\phi(o_i|q)} \right] = -\frac{\alpha}{N} \sum_{o_i} \pi_\phi(o_i|q) \cdot \log \frac{\pi_\phi(o_i|q)}{\pi_\theta(o_i|q)} = -\frac{\alpha}{N} \cdot \mathbb{D}_{\text{KL}} \left(\pi_\phi(\cdot|q) \| \pi_\theta(\cdot|q) \right).$$

Substituting back, we obtain

$$J_q(\phi) = \mathbb{E}_{o_i \sim \pi_\phi(\cdot|q)} \left[r(q, o_i) \right] - \frac{\alpha}{N} \mathbb{D}_{\text{KL}} \left(\pi_\phi(\cdot|q) \| \pi_\theta(\cdot|q) \right).$$

Both terms are differentiable in ϕ, and so taking the gradient and applying Theorem F.1, we have

$$\eta_{\text{eff}} \cdot \mathbb{E} \left[\nabla_\phi \mathcal{L}_{\texttt{Small}}(\phi) \;\middle|\; q \right] = \eta \cdot \nabla_\phi \mathbb{E}_{o_i \sim \pi_\phi(\cdot|q)} \left[r(q, o_i) \right] - \eta \cdot \frac{\alpha}{N} \nabla_\phi \mathbb{D}_{\text{KL}} \left(\pi_\phi(\cdot|q) \| \pi_\theta(\cdot|q) \right),$$

which completes the proof. □

When $\alpha > 0$, the on-policy KD reward introduces an additional direction towards the large model, whereas with $\alpha = 0$ we would only have the direction towards maximizing the original reward. This is similar in spirit to (Xu et al., 2025a), but adapted for the effective reward setting and rigorously derived.