LEARNING BEYOND TEACHER: GENERALIZED ON-POLICY DISTILLATION WITH REWARD EXTRAPOLATION

Anonymous authors
Paper under double-blind review

ABSTRACT

On-policy distillation (OPD), which aligns the student with the teacher's logit distribution on student-generated trajectories, has demonstrated strong empirical gains in improving student performance and often outperforms off-policy distillation and reinforcement learning (RL) paradigms. In this work, we first theoretically show that OPD is a special case of dense KL-constrained RL where the reward function and the KL regularization are always weighted equally and the reference model can by any model. Then, we propose the **Generalized On-Policy Distillation** (**G-OPD**) framework, which extends the standard OPD objective by introducing a flexible reference model and a reward scaling factor that controls the relative weight of the reward term against the KL regularization. Through comprehensive experiments on math reasoning and code generation tasks, we derive two novel insights: (1) Setting the reward scaling factor to be greater than 1 (i.e., *reward extrapolation*), which we term **ExOPD**, consistently improves over standard OPD across a range of teacher-student size pairings. In particular, in the setting where we merge the knowledge from different domain experts, obtained by applying domain-specific RL to the same student model, back into the original student, **ExOPD enables the student to even surpass the teacher's performance boundary and outperform the domain teachers**. (2) Building on ExOPD, we further find that in the strong-to-weak distillation setting (i.e., distilling a smaller student from a larger teacher), performing *reward correction* by choosing the reference model as the teacher's base model before RL yields a more accurate reward signal and further improves distillation performance.

1 INTRODUCTION

Recently, on-policy distillation (OPD) (Agarwal et al., 2024; Yang et al., 2025a; Lu & Lab, 2025) has emerged as an effective post-training paradigm for improving capabilities of Large Language Models (LLMs). Unlike prior off-policy distillation methods (Taori et al., 2023; Guha et al., 2025) that train the student on teacher-generated trajectories, OPD allows the student to learn from the teacher's supervision (i.e., predicted logits) on *student*-generated tokens. Previous studies have shown that OPD can not only serve as a promising multi-task post-training paradigm to near-losslessly merge the capabilities acquired by different RL variants across domains back into the original base model (Xiao et al., 2026; Xu et al., 2026), but also be effective and efficient in distilling the capabilities of a larger teacher into a smaller student (Gu et al., 2024; Yang et al., 2025a).

Despite its empirical effectiveness, a mechanistic understanding of OPD remains limited in the field, leaving its full potential under-explored. In this work, we bridge this gap by establishing a theoretical connection between OPD and dense reinforcement learning (RL), and by extending standard OPD into a generalized formulation.

First, we make derivations to show that OPD is essentially a special case of the standard dense RL with Kullback–Leibler (KL) constraint, where the token-level reward function is always weighted equally with the KL regularization and the reference model can be chosen arbitrarily. Building on this insight, we generalize the OPD objective to a more universal formulation by further introducing a *reward scaling factor* that controls the relative weight of the reward term against the KL regular-

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.

ization, in addition to the *flexible reference model*. We refer to this generalized formulation as the **Generalized On-Policy Distillation** (**G-OPD**) framework.

Based on the G-OPD framework, we theoretically analyze how the reward scaling factor and the choice of reference model affect distillation effectiveness across different settings, supported by comprehensive experiments in both math reasoning and code generation domains. In the first setting, the teacher is obtained by applying domain-specific RL to the student, and the reference model is naturally fixed to the student's initial state. We show that (1) when the reward scaling factor lies in $(0, 1)$ (i.e., *reward interpolation*), the distilled student exhibits behaviors (e.g., performance and response length) that fall between the reference and teacher models; (2) when the reward scaling factor is greater than 1 (i.e., *reward extrapolation*), the student can learn beyond the teacher's capability boundary and outperform teacher in domain tasks. We refer to the reward extrapolation variant as **ExOPD**. We further show that ExOPD extends well to the multi-teacher distillation setting, **enabling a unified student to surpass all domain teachers**. Second, we study the strong-to-weak distillation setting, where a smaller student is distilled from a larger teacher. In this setting, we demonstrate that replacing the reference model from the student's initial policy to the teacher's pre-RL variant (i.e., *reward correction*) in ExOPD yields a more accurate reward signal and further improves distillation performance. However, the limitations of this practice are that it assumes access to the teacher's pre-RL variant and incurs more computational cost on computing the log-probabilities of the larger reference model. Despite these limitations, ExOPD and ExOPD with reward correction significantly outperform standard OPD in the strong-to-weak distillation setting.

2 Related Work

Off-Policy Distillation. Knowledge distillation (KD) (Hinton et al., 2015) is a widely used technique for transferring knowledge from a domain expert (teacher) to a student model. Most prior studies focus on *off-policy* distillation, where the student is trained on trajectories generated by the teacher, either by aligning the student's logits distribution with the teacher's via a Kullback–Leibler (KL) divergence loss on token logits (Sanh et al., 2019; Kim & Rush, 2016; Guo et al., 2025), or by directly performing supervised fine-tuning (SFT) with a cross-entropy loss on the teacher-generated tokens (Taori et al., 2023; Zhou et al., 2023; Guha et al., 2025). This practice has been shown to effectively improve the student model across a broad range of capabilities (Ding et al., 2023; Yang et al., 2025d; Ye et al., 2025b) in the LLM era.

On-Policy Distillation. By sampling trajectories from the student and aligning the student with the teacher's logit distribution on each token of these student-generated trajectories, on-policy distillation (OPD) (Agarwal et al., 2024; Gu et al., 2024) realizes *dense on-policy* learning. Empirically, OPD has been shown to achieve faster and more effective distillation than off-policy distillation (Yang et al., 2025a; Lu & Lab, 2025). Recent OPD studies have explored distillation across different model families (Patiño et al., 2025), developed black-box on-policy distillation methods that do not require access to the teacher's logits (Ye et al., 2025a), and investigated the self-distillation paradigm that leverages the LLM's in-context capabilities to distill textual context information into its parameters (Yang et al., 2025b; Hübotter et al., 2026; Shenfeld et al., 2026; Zhao et al., 2026; Penaloza et al., 2026).

3 Methodology

3.1 Preliminaries

Off-Policy Distillation. Let D denote the input distribution, and let $\pi_{\boldsymbol{\theta}}$ and π^* denote the student and teacher policies, respectively. The general form of Knowledge Distillation (KD) (Hinton et al., 2015) can be written as

$$\mathcal{J}_{\mathrm{KD}}(\boldsymbol{\theta}) = \min_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{x}\sim D, \boldsymbol{y}\sim\pi^*(\cdot|\boldsymbol{x})}\Big[\mathcal{D}_{\mathrm{KL}}\big(\pi^*(\boldsymbol{y}|\boldsymbol{x}) \,\big\|\, \pi_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})\big)\Big], \tag{1}$$

where $\mathcal{D}_{\mathrm{KL}}$ denotes the Kullback–Leibler (KL) divergence loss. **The major drawback of this paradigm is its *off-policy* nature**: the student is trained to imitate the teacher's behavior, rather than to learn from reward signals induced by its own actions. As a result, it may fail to adapt and generalize from its own experience at test time, when faced with similar problems.

On-Policy RL. We use $\pi_{\boldsymbol{\theta}}$ to denote the policy model. The RL objective can be formulated as

$$\mathcal{J}_{\text{RL}}(\boldsymbol{\theta}) = \max_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{x}\sim D, \boldsymbol{y}\sim\pi_{\boldsymbol{\theta}}(\cdot|x)}\Big[r(\boldsymbol{x},\boldsymbol{y}) - \beta\mathcal{D}_{\text{KL}}(\pi_{\boldsymbol{\theta}} \,\|\, \pi_{\text{ref}})\Big]. \tag{2}$$

The trajectories $\boldsymbol{y}$ are sampled from the current policy model, making the training remain *on-policy*. $r(\boldsymbol{x},\boldsymbol{y})$ is the reward function. To solve Eq. (2), a common approach is to apply policy gradient (Sutton et al., 1998), updating the policy parameters using an estimated gradient of the form

$$\nabla_{\boldsymbol{\theta}}\mathcal{J}_{\text{RL}}(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{x}\sim D,\ \boldsymbol{y}\sim\pi_{\boldsymbol{\theta}}(\cdot|\boldsymbol{x})}\Big[\sum_{t=1}^{T} A_t \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(y_t|\boldsymbol{x},\boldsymbol{y}_{<t})\Big], \tag{3}$$

where A_t is the relative advantage of token y_t over a baseline value. In practice, **the reward signal in RL is often sparse**: the policy model only receives a reward at the final token after the response is completed, which may make optimization inefficient and ineffective (Cui et al., 2025).

On-Policy Distillation. On-Policy Distillation (OPD) (Agarwal et al., 2024; Gu et al., 2024; Lu & Lab, 2025) inherits the *on-policy* nature of policy training and the advantage of *dense credit assignment*, making it an efficient post-training paradigm (Yang et al., 2025a; Xiao et al., 2026). The main idea of OPD is to let the student generate its own trajectories, and then minimize the reverse KL divergence between the student and the teacher π^* on those student-generated trajectories:

$$\mathcal{J}_{\text{OPD}}(\boldsymbol{\theta}) = \min_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{x}\sim D,\boldsymbol{y}\sim\pi_{\boldsymbol{\theta}}(\cdot|\boldsymbol{x})}\Big[\mathcal{D}_{\text{KL}}\Big(\pi_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x}) \,\Big\|\, \pi^*(\boldsymbol{y}|\boldsymbol{x})\Big)\Big]. \tag{4}$$

Notice that in Eq. (4), the trajectories $\boldsymbol{y}$ are generated by the policy model itself, resulting in the on-policy training. Also, we can get the gradient of OPD as (detailed derivations are in Appendix A)

$$\nabla_{\boldsymbol{\theta}}\mathcal{J}_{\text{OPD}}(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{x}\sim D,\boldsymbol{y}\sim\pi_{\boldsymbol{\theta}}(\cdot|\boldsymbol{x})}\Big[\sum_{t=1}^{T}\Big(\sum_{t'=t}^{T}\big(\log\pi_{\boldsymbol{\theta}}(y_{t'}|\boldsymbol{x},\boldsymbol{y}_{<t'}) - \log\pi^*(y_{t'}|\boldsymbol{x},\boldsymbol{y}_{<t'})\big)\Big)\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(y_t|\boldsymbol{x},\boldsymbol{y}_{<t})\Big].$$

In practice, current studies (Lu & Lab, 2025; Xiao et al., 2026) use a discount factor of 0 (focus on next-token optimization only) and approximate the gradient as

$$\nabla_{\boldsymbol{\theta}}\mathcal{J}_{\text{OPD}}(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{x}\sim D,\boldsymbol{y}\sim\pi_{\boldsymbol{\theta}}(\cdot|\boldsymbol{x})}\Big[\sum_{t=1}^{T}\big(\log\pi_{\boldsymbol{\theta}}(y_t|\boldsymbol{x},\boldsymbol{y}_{<t}) - \log\pi^*(y_t|\boldsymbol{x},\boldsymbol{y}_{<t})\big)\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(y_t|\boldsymbol{x},\boldsymbol{y}_{<t})\Big].$$

Comparing with Eq. (3), we can see that $-\big(\log\pi_{\boldsymbol{\theta}}(y_t|\boldsymbol{x},\boldsymbol{y}_{<t}) - \log\pi^*(y_t|\boldsymbol{x},\boldsymbol{y}_{<t})\big)$ can be regarded as the token-level advantage, thereby providing dense credit assignment for each token-level action.

3.2 Generalized On-Policy Distillation

In this section, we first start from Eq. (4) and derive a generalized formulation of OPD:

$$\begin{aligned}
\mathcal{J}_{\text{OPD}}(\boldsymbol{\theta}) &= \min_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{x}\sim D,\boldsymbol{y}\sim\pi_{\boldsymbol{\theta}}(\cdot|\boldsymbol{x})}\Big[\log\pi_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x}) - \log\pi^*(\boldsymbol{y}|\boldsymbol{x})\Big] \\
&= \max_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{x}\sim D,\boldsymbol{y}\sim\pi_{\boldsymbol{\theta}}(\cdot|\boldsymbol{x})}\Big[\log\pi^*(\boldsymbol{y}|\boldsymbol{x}) - \log\pi_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})\Big] \\
&= \max_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{x}\sim D,\boldsymbol{y}\sim\pi_{\boldsymbol{\theta}}(\cdot|\boldsymbol{x})}\Big[\big(\log\pi^*(\boldsymbol{y}|\boldsymbol{x}) - \log\pi_{\text{ref}}(\boldsymbol{y}|\boldsymbol{x})\big) - \big(\log\pi_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x}) - \log\pi_{\text{ref}}(\boldsymbol{y}|\boldsymbol{x})\big)\Big] \\
&= \max_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{x}\sim D,\boldsymbol{y}\sim\pi_{\boldsymbol{\theta}}(\cdot|\boldsymbol{x})}\Big[\log\frac{\pi^*(\boldsymbol{y}|\boldsymbol{x})}{\pi_{\text{ref}}(\boldsymbol{y}|\boldsymbol{x})} - \mathcal{D}_{\text{KL}}\big(\pi_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x}) \,\|\, \pi_{\text{ref}}(\boldsymbol{y}|\boldsymbol{x})\big)\Big].
\end{aligned}$$

Therefore, we have the following remark to establish the connection between OPD and RL:

Remark

By introducing a third reference model π_{ref}, the OPD objective in Eq. (4) becomes equivalent to a specific KL-constrained RL objective in Eq. (2), where the reward function $r(\boldsymbol{x},\boldsymbol{y}) = \log\frac{\pi^*(\boldsymbol{y}|\boldsymbol{x})}{\pi_{\text{ref}}(\boldsymbol{y}|\boldsymbol{x})}$, the KL regularization is applied between the policy model $\pi_{\boldsymbol{\theta}}$ and the reference model π_{ref}, and the reward and KL terms are weighted equally (i.e., $\beta = 1$ in Eq. (2)).

However, we emphasize that OPD differs from standard RL in the following key respects:

(1) Dense rewards. As discussed above, in standard RL the model typically receives an effective reward only at the final token, while the rewards for all other tokens are zero:

$$r_t^{RL} = \begin{cases} 0 & t = 1, \cdots, T-1, \\ \text{Outcome Reward} & t = T. \end{cases} \tag{5}$$

However, in OPD, each token-level action receives an effective reward

$$r_t^{OPD} = \log \frac{\pi^*(y_t|\boldsymbol{x}, \boldsymbol{y}_{<t})}{\pi_{\text{ref}}(y_t|\boldsymbol{x}, \boldsymbol{y}_{<t})}, \quad t = 1, \cdots, T. \tag{6}$$

This token-level reward takes essentially the same form as the *implicit reward* defined in Rafailov et al. (2023). Implicit reward is initially derived from the closed-form solution of Eq. (2), which can be written as

$$r(\boldsymbol{x}, \boldsymbol{y}) = \beta \log \frac{\pi_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})}{\pi_{\text{ref}}(\boldsymbol{y}|\boldsymbol{x})} + \beta \log Z(\boldsymbol{x}), \text{ where } Z(\boldsymbol{x}) = \sum_{\boldsymbol{y}} \pi_{\text{ref}}(\boldsymbol{y}|\boldsymbol{x}) \exp(\frac{1}{\beta} r(\boldsymbol{x}, \boldsymbol{y})). \tag{7}$$

As we can see, since $\log Z(\boldsymbol{x})$ is a constant depending only on $\boldsymbol{x}$, $\log \frac{\pi_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})}{\pi_{\text{ref}}(\boldsymbol{y}|\boldsymbol{x})}$ can be regarded as a well-defined proxy of the true reasoning reward, and this idea is adopted in previous studies (Yuan et al., 2024; Cui et al., 2025; Yang et al., 2025c; Liu et al., 2025b) to provide dense supervision for RL. However, in OPD, the implicit reward $\log \frac{\pi^*(\boldsymbol{y}|\boldsymbol{x})}{\pi_{\text{ref}}(\boldsymbol{y}|\boldsymbol{x})}$ does not require π^* to be obtained by applying RL starting from π_{ref}. In fact, π^* and π_{ref} can even be models of different sizes. Nevertheless, **this reward function still captures the log-probability shift from the reference (π_{ref}) distribution to the expert (π^*) distribution**, and thus provides a meaningful training signal.

(2) Fixed weighting between the reward function and the KL regularization. As revealed in the remark, in OPD, the reward term and the KL regularization are always weighted equally. In what follows, we present and discuss our generalized OPD formulation by introducing a reward scaling factor that allows us to adjust the relative weight of the reward term against the KL regularization.

(3) Flexible choice of the reference model. In RL (i.e., Eq. (2)), the reference model is typically initialized as the policy model's starting checkpoint. However, we note that in OPD (i.e., Eq. (8)), the introduced reference model can be any model, since this choice does not affect the final simplification of the objective back to its original form in Eq. (4). In what follows, we discuss how different choices of π_{ref} affect our proposed generalized OPD framework. By default, the reference model is selected as the student's initial policy.

From the above discussion, we can see that OPD offers two key advantages over RL—dense reward signals and a flexible choice of reference model—yet it fixes the relative weighting between the reward function and the KL regularization to $1:1$. This motivates us to follow Eq. (2) and generalize the original OPD objective in Eq. (4) into a general dense RL objective with a flexible KL constraint, by introducing both a third reference model and an additional **reward scaling factor** λ:

$$\mathcal{J}_{\text{G-OPD}}(\boldsymbol{\theta}) = \max_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{x}\sim D, \boldsymbol{y}\sim\pi_{\boldsymbol{\theta}}(\cdot|\boldsymbol{x})} \left[\lambda \log \frac{\pi^*(\boldsymbol{y}|\boldsymbol{x})}{\pi_{\text{ref}}(\boldsymbol{y}|\boldsymbol{x})} - \mathcal{D}_{\text{KL}}(\pi_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x}) \,\|\, \pi_{\text{ref}}(\boldsymbol{y}|\boldsymbol{x})) \right]. \tag{8}$$

The above Eq. (8) presents our **Generalized On-Policy Distillation (G-OPD)** formulation, where λ controls the relative weight of the reward against the KL regularization, and is essential $\frac{1}{\beta}$ in Eq. (2). As we can see, compared to RL, G-OPD enables dense credit assignment and a more flexible choice of reference model; compared to OPD, it further allows more general control over the reward weight. In the following, we discuss in detail about the two crucial components, λ and π_{ref}, in G-OPD.

Reward interpolation and extrapolation. The optimal solution to G-OPD in Eq. (8) satisfies that

$$\begin{aligned} \log \pi_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x}) &= \lambda \log \pi^*(\boldsymbol{y}|\boldsymbol{x}) + (1-\lambda) \log \pi_{\text{ref}}(\boldsymbol{y}|\boldsymbol{x}) \\ &= \log \pi^*(\boldsymbol{y}|\boldsymbol{x}) + (\lambda - 1)(\log \pi^*(\boldsymbol{y}|\boldsymbol{x}) - \log \pi_{\text{ref}}(\boldsymbol{y}|\boldsymbol{x})). \end{aligned} \tag{9}$$

This reveals that, (1) when $0 < \lambda < 1$, G-OPD encourages the student model's log-probability distribution to match a linear interpolation between that of the teacher and reference models. This can also be interpreted as replacing the reward r with $\lambda \cdot r + (1-\lambda) \cdot 0$. Therefore, we refer to this case as ***reward interpolation***. We conjecture that, under this setting, the student trained with G-OPD may exhibit behavior (e.g., performance, response length, etc.) that lies between the reference

model and the standard OPD with $\lambda = 1$. (2) When $\lambda > 1$, G-OPD encourages the student's log-probability distribution to go beyond matching the teacher's log-probabilities by additionally fitting an extra shift term $(\lambda - 1)(\log \pi^* - \log \pi_{\text{ref}})$. From the perspective of rewards, G-OPD with $\lambda > 1$ performs an extrapolation of the reward function's weight; thus, we refer to this regime as ***reward extrapolation***. We wonder whether reward extrapolation can outperform standard OPD, and in a special case, when the teachers are domain experts obtained by applying RL to the same student (Xiao et al., 2026) in different domains, **can reward extrapolation in G-OPD distill a unified student that surpasses all the domain teachers?**

Reward correction in strong-to-weak distillation. When the reward scaling factor $\lambda \neq 1$, different choices of the reference model π_{ref} in Eq. (8) lead to different objectives. We discuss the choices of π_{ref} in two cases: (1) One application of G-OPD is to merge the capabilities of several experts, each obtained by applying domain-specific RL on the same base model, back into the original base model (Xiao et al., 2026). In this setting, π_{ref} is naturally chosen as the original base model, and the reward function in G-OPD is exactly the implicit reward defined in Eq. (7). (2) Another distillation setting is *strong-to-weak distillation* (Yang et al., 2025a), i.e., distilling a large teacher into a smaller student. In this case, π_{ref} admits two choices: (i) the student's base model, $\pi_{\text{base}}^{\text{student}}$, which is the default setting; or (ii) the teacher expert's pre-RL base model, $\pi_{\text{base}}^{\text{teacher}}$, assuming it is available. To compare these two choices, we first rewrite the G-OPD objective into an equivalent form:

$$\begin{aligned}\mathcal{J}_{\text{G-OPD}}(\boldsymbol{\theta}) &= \max_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{x}\sim D, \boldsymbol{y}\sim\pi_{\boldsymbol{\theta}}(\cdot|\boldsymbol{x})}\left[\lambda \log \frac{\pi^*(\boldsymbol{y}|\boldsymbol{x})}{\pi_{\text{ref}}(\boldsymbol{y}|\boldsymbol{x})} - \mathcal{D}_{\text{KL}}\big(\pi_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x}) \,\big\|\, \pi_{\text{ref}}(\boldsymbol{y}|\boldsymbol{x})\big)\right] \\ &= \max_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{x}\sim D, \boldsymbol{y}\sim\pi_{\boldsymbol{\theta}}(\cdot|\boldsymbol{x})}\left[(\lambda-1) \log \frac{\pi^*(\boldsymbol{y}|\boldsymbol{x})}{\pi_{\text{ref}}(\boldsymbol{y}|\boldsymbol{x})} - \mathcal{D}_{\text{KL}}\big(\pi_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x}) \,\big\|\, \pi^*(\boldsymbol{y}|\boldsymbol{x})\big)\right].\end{aligned} \tag{10}$$

Now, under the same KL regularization strength, we can see that choosing $\pi_{\text{ref}} = \pi_{\text{base}}^{\text{teacher}}$ is more reasonable. The reason is that the reward $\log \frac{\pi^*}{\pi_{\text{base}}^{\text{teacher}}}$ corresponds to the implicit reward induced by the teacher's RL post-training, and is thus well-defined according to Eq. (7). In contrast, $\log \frac{\pi^*}{\pi_{\text{base}}^{\text{student}}}$ can be noisier, since there exists fundamental gap between the internal knowledge and capacity of teacher and student base models. Therefore, in the strong-to-weak distillation setting, we think that applying a ***reward correction*** to the default reward $\log \frac{\pi^*}{\pi_{\text{base}}^{\text{student}}}$—by adding $\log \frac{\pi_{\text{base}}^{\text{student}}}{\pi_{\text{base}}^{\text{teacher}}}$ to obtain $\log \frac{\pi^*}{\pi_{\text{base}}^{\text{teacher}}}$—can lead to better distillation performance. The limitations, however, are that this requires access to $\pi_{\text{base}}^{\text{teacher}}$ and incurs additional computation, since computing $\log \pi_{\text{base}}^{\text{teacher}}$ requires more cost than computing $\log \pi_{\text{base}}^{\text{student}}$.

Remark

By introducing a reference model π_{ref} and a reward scaling factor λ, we formulate the Generalized On-Policy Distillation framework as

$$\mathcal{J}_{\text{G-OPD}}(\boldsymbol{\theta}) = \max_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{x}\sim D, \boldsymbol{y}\sim\pi_{\boldsymbol{\theta}}(\cdot|\boldsymbol{x})}\left[\lambda \log \frac{\pi^*(\boldsymbol{y}|\boldsymbol{x})}{\pi_{\text{ref}}(\boldsymbol{y}|\boldsymbol{x})} - \mathcal{D}_{\text{KL}}\big(\pi_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x}) \,\big\|\, \pi_{\text{ref}}(\boldsymbol{y}|\boldsymbol{x})\big)\right].$$

(1) For $0 < \lambda < 1$, G-OPD yields a student whose behavior lies between that of the reference model and that of the student trained by standard OPD (i.e., $\lambda = 1$). In contrast, $\lambda > 1$ can potentially deliver larger gains than standard OPD, and may even produce a student that outperforms the teacher in certain cases. Note that setting $\lambda \neq 1$ incurs additional computational cost on computing $\log \pi_{\text{ref}}$.
(2) In strong-to-weak distillation, choosing the teacher's pre-RL model $\pi_{\text{base}}^{\text{teacher}}$ as the reference may yield better distillation performance. However, this comes with two limitations: it requires access to an additional model $\pi_{\text{base}}^{\text{teacher}}$, and it increases computational cost because computing $\log \pi_{\text{base}}^{\text{teacher}}$ requires more computational cost than computing $\log \pi_{\text{base}}^{\text{student}}$.

Finally, the approximated gradient of G-OPD can be written as

$$\nabla_{\boldsymbol{\theta}} \mathcal{J}_{\text{G-OPD}}(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{x}\sim D, \boldsymbol{y}\sim\pi_{\boldsymbol{\theta}}(\cdot|\boldsymbol{x})}\left[\sum_{t=1}^{T} A_t^{\text{G-OPD}} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(y_t|\boldsymbol{x}, \boldsymbol{y}_{<t})\right], \tag{11}$$

where $A_t^{\text{G-OPD}} = \big(\log \pi_{\boldsymbol{\theta}}(y_t|\boldsymbol{x}, \boldsymbol{y}_{<t}) - \log \pi^*(y_t|\boldsymbol{x}, \boldsymbol{y}_{<t})\big) + (\lambda - 1)\big(\log \pi_{\text{ref}}(y_t|\boldsymbol{x}, \boldsymbol{y}_{<t}) - \log \pi^*(y_t|\boldsymbol{x}, \boldsymbol{y}_{<t})\big)$.

4 Experiments and Analysis

In this section, we conduct a series of extensive experiments on math reasoning and code generation tasks to analyze the properties of the proposed G-OPD framework and assess the effectiveness of ExOPD. We begin with preliminary experiments on same-size teacher-student pairs in Section 4.1.2, where we investigate the impact of the reward scaling factor within G-OPD. We then explore the effectiveness of ExOPD in the multi-teacher distillation setting in Section 4.1.3. Finally, we present experimental results in the strong-to-weak distillation setting in Section 4.2.

4.1 Experiments with Same-Sized Student and Teacher

Here, we consider the scenario where the domain teachers are reinforced models derived from the student through domain-specific RL.

4.1.1 Experimental Settings

Base Model. We conduct experiments using Qwen3-4B-Non-Thinking (Yang et al., 2025a) as the student model. The domain teachers are derived by applying RL to it on domain-specific data.

Training Datasets. We filter the DeepMath (He et al., 2025) dataset to select 57K samples with a difficulty level greater than or equal to 6 to form the math RL data, and use Eurus-RL-Code (Cui et al., 2025) as the code RL data, which consists of 25K samples. We then apply RL to the base model on two datasets separately to get domain teachers, Qwen3-4B-Non-Thinking-RL-Math and Qwen3-4B-Non-Thinking-RL-Code. The distillation data is the same as the RL data.

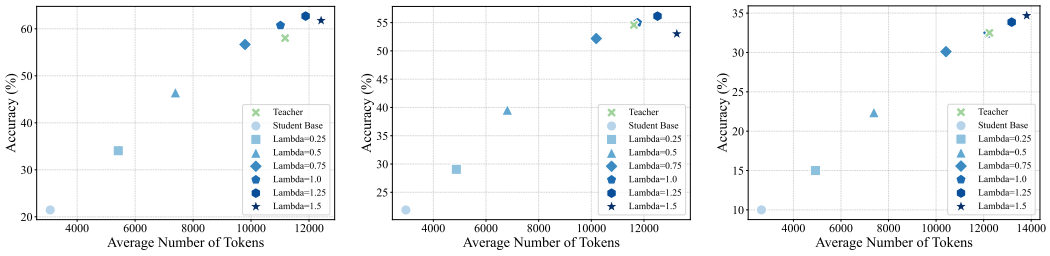
Training Settings. We apply Group Relative Policy Optimization (GRPO) (Shao et al., 2024) to obtain domain teachers. Detailed training hyper-parameters in GRPO are in Appendix B. After this, we implement G-OPD on the original student model (i.e., Qwen3-4B-Non-Thinking) with different reward scaling factors $\lambda \in \{0.0, 0.25, 0.5, 0.75, 1.0, 1.25, 1.5\}$. Note that $\lambda = 0.0$ corresponds to the initial state Qwen3-4B-Non-Thinking, and $\lambda = 1.0$ corresponds to standard OPD. The reference model here is fixed naturally as Qwen3-4B-Non-Thinking. Detailed training hyper-parameters in G-OPD are in Appendix B. More details are in Appendix B.

Evaluation. For the evaluation of math reasoning, we select four competition-level benchmarks: AIME24 (AI-MO, 2024), AIME25 (OpenCompass, 2025), HMMT25 (February) (Balunović et al., 2025), and HMMT25 (November) (Balunović et al., 2025). For the evaluation of code generation, we select three test sets: HumanEval+, MBPP+ (Liu et al., 2023), and LiveCodeBench (v6 only, February 2025∼May 2025) (Jain et al., 2024). In all evaluations, we set the temperature to 1.0, top-p to 1.0, and the maximum generation length to 16,384. On each math reasoning benchmark, we sample 32 solutions for each problem; whereas each code generation benchmark, we sample 4 solutions per problem. We then report the average accuracy of each model on each benchmark. We adopt `Math-Verify` as a rule-based verifier to validate answer correctness for math reasoning.

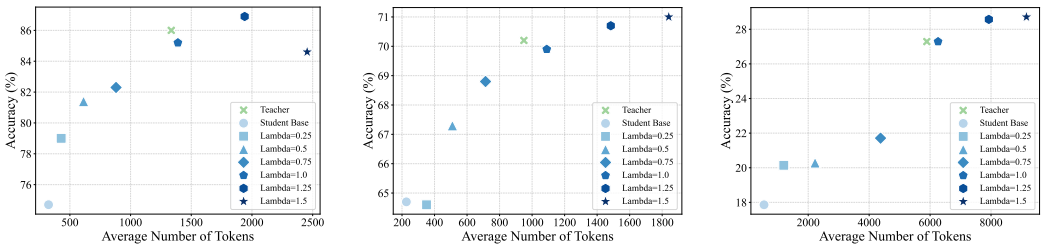
4.1.2 Results of Single-Teacher Distillation

We first explore the impact of reward scaling factor λ in G-OPD in the same-sized single-teacher distillation setting as the preliminary experiments. We show the selected results in Figure 1, while we leave more visualizations in Figure 3 and Figure 4.

We can draw the following conclusions: (1) Standard OPD can fully recover the post-training behavior, which is consistent with previous findings. (2) Reward interpolation ($0 < \lambda < 1$) produces a student whose behavior (performance and response length) lies between the base model and the teacher model. Also, both the performance and response length increase monotonically as λ grows, approaching the behavior of the teacher. This property can be leveraged to achieve budget-controlled reasoning (Yang et al., 2025d; Liang et al., 2026). (3) **Reward extrapolation ($\lambda > 1$) has the potential to produce a student that surpasses the domain teacher.** As observed, ExOPD with appropriate reward extrapolation (i.e., $\lambda = 1.25$) consistently outperforms OPD and the domain



(a) Results on AIME24 (b) Results on AIME25 (c) Results on HMMT25 (Feb.)



(d) Results on HumanEval+ (e) Results on MBPP+ (f) Results on LiveCodeBench

Figure 1: Trends in the average number of tokens and the average accuracy of the on-policy distilled models across different benchmarks under varying reward scaling factors.

teacher (also see Table 1), while excessive reward extrapolation (i.e., $\lambda = 1.5$) may degrade performance. This can be explained by the fact that continuously increasing λ introduces the risk of the student hacking the implicit reward in Eq. (6), by aggressively fitting the peak of the log ratio, even if some tokens have excessively large log ratios due to bias. Furthermore, we can see that the response lengths of the students produced by ExOPD continue to increase, which may be due to the length bias issue of the implicit reward (Yang et al., 2025c).

In Table 7, we show that more continued RL training on the teacher shows smaller improvement compared to ExOPD with fewer steps. We also demonstrate the generalizability and effectiveness of ExOPD when the teachers are trained with sufficient RL steps (i.e., 1200 RL steps), with the corresponding results provided in Appendix E.

4.1.3 Results of Multi-Teacher Distillation

Based on above analysis, we conduct experiments in the multi-teacher distillation setting, where we aim to merge the capabilities from different domain teachers, obtained by applying domain-specific RL to the same base model, into the original base model through OPD (Xiao et al., 2026; Xu et al., 2026). This has been demonstrated to be an effective multi-task post-training paradigm. Specifically, the domain teachers are the above RL variants Qwen3-4B-Non-Thinking-RL-Math/Code, and the student model is Qwen3-4B-Non-Thinking. From the preliminary results in Section 4.1.2, we can see that $\lambda = 1.25$ in ExOPD consistently leads to better performance than OPD. Thus, **in all subsequent experiments, we fix $\lambda = 1.25$ for ExOPD without any further specific tuning**. Besides OPD, we also compare against two baselines: (1) **Supervised fine-tuning (SFT)**, which trains the student on the teachers' generated trajectories via Cross-Entropy Loss. We ensure that the number of trajectories used for SFT is consistent with those in OPD and ExOPD. More details can be found in Appendix B. (2) **ExPO** (Zheng et al., 2025), a weight extrapolation method. We implement ExPO by first averaging the weights of all domain teachers, then extrapolating the weights against the student model using an extrapolation factor α, which is tuned from $\{0.25, 0.5\}$ following the recommendations. For a fair comparison, we downweight the sample size of the math RL data to match that of the code RL data in both OPD and ExOPD here, ensuring that each domain has the same sample size.

The results of multi-teacher distillation are shown in Table 1. As we can see, SFT produces a sub-optimal student, while the performance ceiling of OPD is typically bounded by the teachers. ExPO, though training-free, cannot ensure that the weight-extrapolated student consistently surpasses all

Table 1: Comparison against off-policy distillation (SFT) and weight extrapolation (ExPO) methods the multi-teacher setting with same-sized teacher-student pairs. Each numerical subscript indicates the absolute **improvement** or **degradation** **compared to the domain teacher model**.

Method	Math Reasoning					Code Generation			
	AIME24	AIME25	HMMT-25(Feb.)	HMMT-25(Nov.)	Avg.	Human-Eval+	MBPP+	LCB	Avg.
Teacher	58.0	54.6	32.5	38.9	46.0	86.0	70.2	27.3	61.2
Student	21.5	21.9	10.0	8.0	15.4	74.7	64.7	17.9	52.4
SFT	$58.5_{+0.5}$	$53.3_{-1.3}$	$30.7_{-1.8}$	$34.8_{-4.1}$	$44.3_{-1.7}$	$86.4_{+0.4}$	$69.6_{-0.6}$	$26.4_{-0.9}$	$60.8_{-0.4}$
ExPO	$57.5_{-0.5}$	$54.5_{-0.1}$	$31.7_{-0.8}$	$36.3_{-2.6}$	$45.0_{-1.0}$	$\mathbf{86.7}_{+0.7}$	$\mathbf{72.0}_{+1.8}$	$\mathbf{29.0}_{+1.7}$	$\mathbf{62.6}_{+1.4}$
OPD	$60.6_{+2.6}$	$54.1_{-0.5}$	$32.5_{+0.0}$	$38.3_{-0.6}$	$46.4_{+0.4}$	$84.6_{-1.4}$	$69.5_{-0.7}$	$27.6_{+0.3}$	$60.6_{-0.6}$
ExOPD	$\mathbf{61.0}_{+3.0}$	$\mathbf{56.0}_{+1.4}$	$\mathbf{34.4}_{+1.9}$	$\mathbf{39.2}_{+0.3}$	$\mathbf{47.7}_{+1.7}$	$86.3_{+0.3}$	$70.6_{+0.4}$	$\mathbf{29.0}_{+1.7}$	$62.0_{+0.8}$

domain teachers, lacking good controllability. However, **our method ExOPD consistently outperforms OPD and is the only method that produces a unified student capable of surpassing both domain teachers on all benchmarks**.

Furthermore, we analyze the training dynamics of ExOPD compared to OPD to gain a deeper understanding of ExOPD. We put the comparison in Figure 5. ExOPD achieves higher training rewards but makes the student generate longer response lengths, which is consistent with the evaluation results shown in Figure 1. We also observe that the response entropy of the student trained by ExOPD is higher than that trained by OPD. We attribute this to the fact that the former tends to generate longer responses, increasing the response diversity.

4.2 Experiments in the Strong-to-Weak Distillation Setting

Another practical usage of OPD is for strong-to-weak distillation (Yang et al., 2025a), i.e., distilling capabilities from a larger teacher into a smaller student. In this section, we explore the effectiveness of ExOPD and the additional reward correction practice in the strong-to-weak distillation setting.

4.2.1 Experimental Settings

We select Qwen3-30B-A3B-Instruct-2507 as the teacher model and perform distillation on Qwen3-1.7B-Non-Thinking and Qwen3-4B-Non-Thinking, respectively. We conduct experiments in the math reasoning domain, where the training and evaluation datasets are the same as those used in Section 4.1. The training details are in Appendix B. In ExOPD, we first conduct experiments in the default setting (Section 4.2.2), where we assume the availability of only two models: the student base model and the stronger teacher model. Thus, in this default setting, we set the reference model in ExOPD to the student base model. We also explore the effectiveness of the reward correction technique in Section 4.2.3, where we assume extra access to the teacher's pre-RL variant, which serves as the reference model in ExOPD. We compare ExOPD against standard OPD and SFT.

4.2.2 Results of Strong-to-Weak Distillation

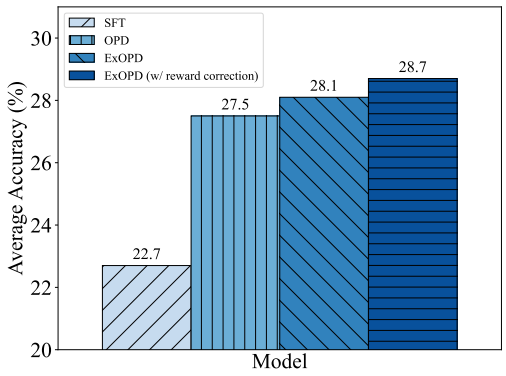
The results in default strong-to-weak distillation setting are put in Table 2. The main conclusion is that **ExOPD can bring significant improvements in strong-to-weak distillation, outperforming off-policy distillation (SFT) and standard OPD by a large margin**. The results reveal that, although the implicit reward $\log \frac{\pi^*}{\pi_{\text{base}}^{\text{student}}}$ may contain noise due to the intrinsic knowledge gap and distribution bias between the small and large models, extrapolating the rewards can still push the limits of OPD in strong-to-weak distillation.

4.2.3 Reward Correction in Strong-to-Weak Distillation

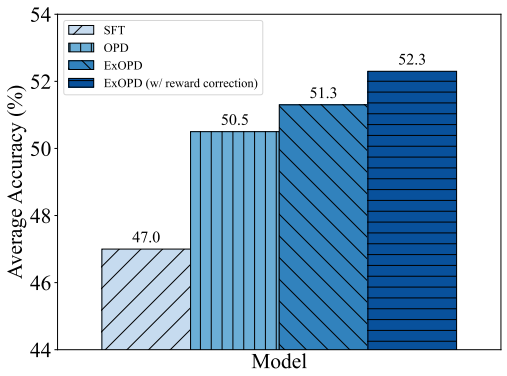
As shown above, the default ExOPD with the reference model fixed as the student base model can already bring significant improvement over OPD. However, as discussed in Remark 3.2, setting the reference model to the teacher's pre-RL variant—if available—may further enhance the distillation performance. Here, we conduct experiments to validate this analysis. Specifically, since we cannot

Table 2: Results on math reasoning in the strong-to-weak distillation setting. The numerical subscript indicates the absolute **improvement** or **degradation** **compared to the standard OPD**.

Method	AIME24	AIME25	HMMT25 (Feb.)	HMMT25 (Nov.)	Avg.
Teacher	74.7	62.8	44.2	57.2	59.7
Student: Qwen3-1.7B-Non-Thinking					
Base	12.3	11.4	6.8	4.5	8.8
SFT	18.1	20.5	9.2	6.3	13.5
OPD	33.0	28.7	15.7	14.9	23.1
ExOPD	**37.3**$_{+4.3}$	**31.5**$_{+2.8}$	**16.2**$_{+0.5}$	**16.5**$_{+1.6}$	**25.4**$_{+2.3}$
Student: Qwen3-4B-Non-Thinking					
Base	21.5	21.9	10.0	8.0	15.4
SFT	45.4	40.9	22.4	31.6	35.1
OPD	55.0	48.0	29.8	37.7	42.6
ExOPD	**58.7**$_{+3.7}$	**50.8**$_{+2.8}$	**33.0**$_{+3.2}$	**38.8**$_{+1.1}$	**45.3**$_{+2.7}$



(a) Results averaged on four **math reasoning** benchmarks, student is Qwen3-1.7B-Non-Thinking, teacher is Qwen3-4B-Non-Thinking-RL-Math



(b) Results averaged on three **code generation** benchmarks, student is Qwen3-1.7B-Non-Thinking, teacher is Qwen3-4B-Non-Thinking-RL-Code

Figure 2: Effect of *reward correction* in the strong-to-weak distillation setting.

get the pre-RL variant of Qwen3-30B-A3B-Instruct-2507, we choose our trained Qwen3-4B-Non-Thinking-RL-Math/Code as the teachers and take Qwen3-4B-Non-Thinking as the pre-RL variant. The student model is Qwen3-1.7B-Non-Thinking.

The comparison results are displayed in Figure 2. The results validate the effectivenss of the reward correction practice, which **consistently boosts the performance of ExOPD**. However, we reiterate that reward correction requires access to $\pi_{\text{base}}^{\text{teacher}}$ and incurs higher computational cost, since it requires computing log-probabilities under a larger reference model than in the default ExOPD.

5 Conclusion

In this work, we conduct an in-depth analysis of the on-policy distillation paradigm. We first establish an interesting connection between OPD and dense KL-constrained RL. Building on this insight, we propose a generalized OPD framework (G-OPD) by introducing (i) a flexible reference model for the implicit reward function and (ii) a reward scaling factor that controls the relative weight of the reward term versus KL regularization. Through comprehensive experiments on math reasoning and code generation tasks, we provide several novel insights: (1) Appropriate reward extrapolation (i.e., setting the reward scaling factor to be larger than 1) can improve OPD performance, and in same-sized multi-teacher distillation it enables learning a unified student that surpasses all domain-specific teachers. We refer to this variant as ExOPD. (2) Moreover, in strong-to-weak distillation, replacing the student's initial policy with the teacher's pre-RL policy as the reference model can further boost the performance of ExOPD.

AI USE STATEMENT

In this work, we used generative AI tools for polishing the writing. However, we have not used generative AI tools for full paper writing. Additionally, we used generative AI tools for code reviewing and debugging. We have reviewed all AI-assisted work. LLM-generated code was verified and tested for correctness by authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work aims to provide an in-depth understanding of the mechanisms underlying on-policy distillation (OPD). We establish a close connection between OPD and dense-credit reinforcement learning (RL), and on this basis, propose a generalized OPD framework and the novel ExOPD algorithm. We hope our work can offer new insights for future research on OPD.

REPRODUCIBILITY STATEMENT

We present the necessary math derivations in Section 3.2 and Appendix A. We provide the detailed training and evaluation settings in Section 4.1.1 and Appendix B. We provide the prompt templates used in the experiments in Appendix G for reproduction. We will open-source the code and checkpoints upon paper acceptance.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *The twelfth international conference on learning representations*, 2024.

AI-MO. Aime 2024. https://huggingface.co/datasets/AI-MO/aimo-validation-aime, 2024.

Mislav Balunović, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. Matharena: Evaluating llms on uncontaminated math competitions, February 2025. URL https://matharena.ai/.

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, et al. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*, 2025.

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. Enhancing chat language models by scaling high-quality instructional conversations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 3029–3051, 2023.

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

Etash Guha, Ryan Marten, Sedrick Keh, Negin Raoof, Georgios Smyrnis, Hritik Bansal, Marianna Nezhurina, Jean Mercat, Trung Vu, Zayne Sprague, et al. Openthoughts: Data recipes for reasoning models. *arXiv preprint arXiv:2506.04178*, 2025.

Yiju Guo, Wenkai Yang, Zexu Sun, Ning Ding, Zhiyuan Liu, and Yankai Lin. Learning to focus: Causal attention distillation via gradient-guided token pruning. *arXiv preprint arXiv:2506.07851*, 2025.

Zhiwei He, Tian Liang, Jiahao Xu, Qiuzhi Liu, Xingyu Chen, Yue Wang, Linfeng Song, Dian Yu, Zhenwen Liang, Wenxuan Wang, et al. Deepmath-103k: A large-scale, challenging, decontaminated, and verifiable mathematical dataset for advancing reasoning. *arXiv preprint arXiv:2504.11456*, 2025.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Jonas Hübotter, Frederike Lübeck, Lejs Behric, Anton Baumann, Marco Bagatella, Daniel Marta, Ido Hakimi, Idan Shenfeld, Thomas Kleine Buening, Carlos Guestrin, et al. Reinforcement learning via self-distillation. *arXiv preprint arXiv:2601.20802*, 2026.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024.

Yoon Kim and Alexander M Rush. Sequence-level knowledge distillation. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pp. 1317–1327, 2016.

Kun Liang, Clive Bai, Xin Xu, Chenming Tang, Sanwoo Lee, Weijie Liu, Saiyong Yang, and Yunfang Wu. Orbit: On-policy exploration-exploitation for controllable multi-budget reasoning. *arXiv preprint arXiv:2601.08310*, 2026.

Jiacai Liu, Yingru Li, Yuqian Fu, Jiawei Wang, Qian Liu, and Yu Shen. When speed kills stability: Demystifying RL collapse from the training-inference mismatch, September 2025a. URL https://richardli.xyz/rl-collapse.

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.

Xiaoqian Liu, Ke Wang, Yuchuan Wu, Fei Huang, Yongbin Li, Junge Zhang, and Jianbin Jiao. Agentic reinforcement learning with implicit step rewards. *arXiv preprint arXiv:2509.19199*, 2025b.

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

OpenCompass. Aime 2025. https://huggingface.co/datasets/opencompass/AIME2025, 2025.

Carlos Miguel Patiño, Kashif Rasul, Quentin Gallouédec, Ben Burtenshaw, Sergio Paniego, Vaibhav Srivastav, Thibaud Frere, Ed Beeching, Lewis Tunstall, Leandro von Werra, and Thomas Wolf. Unlocking on-policy distillation for any model family, 2025.

Emiliano Penaloza, Dheeraj Vattikonda, Nicolas Gontier, Alexandre Lacoste, Laurent Charlin, and Massimo Caccia. Privileged information distillation for language models. *arXiv preprint arXiv:2602.04942*, 2026.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

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.

Idan Shenfeld, Mehul Damani, Jonas Hübotter, and Pulkit Agrawal. Self-distillation enables continual learning. *arXiv preprint arXiv:2601.19897*, 2026.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*, 2024.

Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. Alpaca: A strong, replicable instruction-following model. *Stanford Center for Research on Foundation Models. https://crfm. stanford. edu/2023/03/13/alpaca. html*, 3(6):7, 2023.

Bangjun Xiao, Bingquan Xia, Bo Yang, Bofei Gao, Bowen Shen, Chen Zhang, Chenhong He, Chiheng Lou, Fuli Luo, Gang Wang, et al. Mimo-v2-flash technical report. *arXiv preprint arXiv:2601.02780*, 2026.

Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chenchen Ling, et al. Deepseek-v4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026.

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.

Wenkai Yang, Yankai Lin, Jie Zhou, and Ji-Rong Wen. Distilling rule-based knowledge into large language models. In *Proceedings of the 31st International Conference on Computational Linguistics*, pp. 913–932, 2025b.

Wenkai Yang, Weijie Liu, Ruobing Xie, Yiju Guo, Lulu Wu, Saiyong Yang, and Yankai Lin. Laser: Reinforcement learning with last-token self-rewarding. *arXiv preprint arXiv:2510.14943*, 2025c.

Wenkai Yang, Shuming Ma, Yankai Lin, and Furu Wei. Towards thinking-optimal scaling of test-time compute for llm reasoning. *arXiv preprint arXiv:2502.18080*, 2025d.

Tianzhu Ye, Li Dong, Zewen Chi, Xun Wu, Shaohan Huang, and Furu Wei. Black-box on-policy distillation of large language models. *arXiv preprint arXiv:2511.10643*, 2025a.

Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. Limo: Less is more for reasoning. *arXiv preprint arXiv:2502.03387*, 2025b.

Lifan Yuan, Wendi Li, Huayu Chen, Ganqu Cui, Ning Ding, Kaiyan Zhang, Bowen Zhou, Zhiyuan Liu, and Hao Peng. Free process rewards without process labels. *arXiv preprint arXiv:2412.01981*, 2024.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models. *arXiv preprint arXiv:2601.18734*, 2026.

Chujie Zheng, Ziqi Wang, Heng Ji, Minlie Huang, and Nanyun Peng. Model extrapolation expedites alignment. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1025–1041, 2025.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. Lima: Less is more for alignment. *Advances in Neural Information Processing Systems*, 36:55006–55021, 2023.

A Detailed Math Derivations

Here, we make mathematical derivations to calculate the expected gradients of OPD objective in Eq. (4).

Since

$$\begin{aligned}\mathcal{J}_{\text{OPD}}(\boldsymbol{\theta}) &= \min_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{x}\sim D, \boldsymbol{y}\sim\pi_{\boldsymbol{\theta}}(\cdot|\boldsymbol{x})}\Big[\mathcal{D}_{\text{KL}}\big(\pi_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x}) \,\big\|\, \pi^*(\boldsymbol{y}|\boldsymbol{x})\big)\Big] \\ &= \min_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{x}\sim D, \boldsymbol{y}\sim\pi_{\boldsymbol{\theta}}(\cdot|\boldsymbol{x})}\Big[\log \pi_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x}) - \log \pi^*(\boldsymbol{y}|\boldsymbol{x})\Big].\end{aligned} \tag{12}$$

We can get

$$\begin{aligned}&\nabla_{\boldsymbol{\theta}}\mathcal{J}_{\text{OPD}}(\boldsymbol{\theta}) \\ &= \nabla_{\boldsymbol{\theta}}\mathbb{E}_{\boldsymbol{x}\sim D, \boldsymbol{y}\sim\pi_{\boldsymbol{\theta}}(\cdot|\boldsymbol{x})}\Big[\log \pi_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x}) - \log \pi^*(\boldsymbol{y}|\boldsymbol{x})\Big] \\ &= \nabla_{\boldsymbol{\theta}}\mathbb{E}_{\boldsymbol{x}}\Big[\sum_{\boldsymbol{y}} \pi_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})\big(\log \pi_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x}) - \log \pi^*(\boldsymbol{y}|\boldsymbol{x})\big)\Big] \\ &= \mathbb{E}_{\boldsymbol{x}}\Big[\nabla_{\boldsymbol{\theta}}\sum_{\boldsymbol{y}} \pi_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})\big(\log \pi_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x}) - \log \pi^*(\boldsymbol{y}|\boldsymbol{x})\big)\Big] \\ &= \mathbb{E}_{\boldsymbol{x}}\Big[\sum_{\boldsymbol{y}} \big(\nabla_{\boldsymbol{\theta}}\pi_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})\big)\big(\log \pi_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x}) - \log \pi^*(\boldsymbol{y}|\boldsymbol{x})\big) + \sum_{\boldsymbol{y}} \pi_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})\Big].\end{aligned} \tag{13}$$

Notice that

$$\begin{aligned}\mathbb{E}_{\boldsymbol{x}}\Big[\sum_{\boldsymbol{y}} \pi_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})\Big] &= \mathbb{E}_{\boldsymbol{x}}\Big[\sum_{\boldsymbol{y}} \pi_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})\frac{\nabla_{\boldsymbol{\theta}}\pi_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})}{\pi_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})}\Big] \\ &= \mathbb{E}_{\boldsymbol{x}}\Big[\sum_{\boldsymbol{y}} \nabla_{\boldsymbol{\theta}}\pi_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})\Big] \\ &= \mathbb{E}_{\boldsymbol{x}}\Big[\nabla_{\boldsymbol{\theta}}\sum_{\boldsymbol{y}} \pi_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})\Big] \\ &= \mathbb{E}_{\boldsymbol{x}}\Big[\nabla_{\boldsymbol{\theta}} 1\Big] \\ &= 0.\end{aligned} \tag{14}$$

Therefore, Eq. (13) can be reduced to

$$\begin{aligned}&\nabla_{\boldsymbol{\theta}}\mathcal{J}_{\text{OPD}}(\boldsymbol{\theta}) \\ &= \mathbb{E}_{\boldsymbol{x}}\Big[\sum_{\boldsymbol{y}} \nabla_{\boldsymbol{\theta}}\pi_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})\big(\log \pi_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x}) - \log \pi^*(\boldsymbol{y}|\boldsymbol{x})\big)\Big] \\ &= \mathbb{E}_{\boldsymbol{x}}\Big[\sum_{\boldsymbol{y}} \pi_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})\big(\log \pi_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x}) - \log \pi^*(\boldsymbol{y}|\boldsymbol{x})\big)\Big] \\ &= \mathbb{E}_{\boldsymbol{x}\sim D, \boldsymbol{y}\sim\pi_{\boldsymbol{\theta}}(\cdot|\boldsymbol{x})}\Big[\big(\log \pi_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x}) - \log \pi^*(\boldsymbol{y}|\boldsymbol{x})\big)\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})\Big] \\ &= \mathbb{E}_{\boldsymbol{x}\sim D, \boldsymbol{y}\sim\pi_{\boldsymbol{\theta}}(\cdot|\boldsymbol{x})}\Big[\sum_{t=1}^{T}\sum_{t'=1}^{T}\big(\log \pi_{\boldsymbol{\theta}}(y_{t'}|\boldsymbol{x}, \boldsymbol{y}_{<t'}) - \log \pi^*(y_{t'}|\boldsymbol{x}, \boldsymbol{y}_{<t'})\big)\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(y_t|\boldsymbol{x}, \boldsymbol{y}_{<t})\Big].\end{aligned} \tag{15}$$

Now let's denote

$$\Delta_{t'} = \big(\log \pi_{\boldsymbol{\theta}}(y_{t'}|\boldsymbol{x}, \boldsymbol{y}_{<t'}) - \log \pi^*(y_{t'}|\boldsymbol{x}, \boldsymbol{y}_{<t'})\big),$$

and consider each term $\mathbb{E}_{\boldsymbol{x}\sim D,\boldsymbol{y}\sim\pi_{\boldsymbol{\theta}}(\cdot|\boldsymbol{x})}\Big[\Delta_{t'}\,\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(y_t|\boldsymbol{x},\boldsymbol{y}_{<t})\Big]$ where $t^{'}<t$:

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{x},\boldsymbol{y}}\Big[\Delta_{t'}\,\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(y_t|\boldsymbol{x},\boldsymbol{y}_{<t})\Big] &= \mathbb{E}_{\boldsymbol{x},\boldsymbol{y}}\Big[\mathbb{E}_{y_t}\Big[\Delta_{t'}\,\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(y_t|\boldsymbol{x},\boldsymbol{y}_{<t})\big|\boldsymbol{x},\boldsymbol{y}_{<t}\Big]\Big]\\
&= \mathbb{E}_{\boldsymbol{x},\boldsymbol{y}}\Big[\Delta_{t'}\mathbb{E}_{y_t}\Big[\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(y_t|\boldsymbol{x},\boldsymbol{y}_{<t})\big|\boldsymbol{x},\boldsymbol{y}_{<t}\Big]\Big]\\
&= \mathbb{E}_{\boldsymbol{x},\boldsymbol{y}}\Big[\Delta_{t'}\mathbb{E}_{y_t\sim\pi_{\boldsymbol{\theta}}(\cdot|\boldsymbol{x},\boldsymbol{y}_{<t})}\big[\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(y_t|\boldsymbol{x},\boldsymbol{y}_{<t})\big]\Big]\\
&= \mathbb{E}_{\boldsymbol{x},\boldsymbol{y}}\Big[\Delta_{t'}\sum_{y_t}\nabla_{\boldsymbol{\theta}}\pi_{\boldsymbol{\theta}}(y_t|\boldsymbol{x},\boldsymbol{y}_{<t})\Big]\\
&= \mathbb{E}_{\boldsymbol{x},\boldsymbol{y}}\Big[\Delta_{t'}\nabla_{\boldsymbol{\theta}}\sum_{y_t}\pi_{\boldsymbol{\theta}}(y_t|\boldsymbol{x},\boldsymbol{y}_{<t})\Big]\\
&= \mathbb{E}_{\boldsymbol{x},\boldsymbol{y}}\Big[\Delta_{t'}\nabla_{\boldsymbol{\theta}}1\Big]\\
&= 0.
\end{aligned} \tag{16}$$

Therefore, Eq. (15) can be reduced to

$$\nabla_{\boldsymbol{\theta}}\mathcal{J}_{\text{OPD}}(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{x}\sim D,\boldsymbol{y}\sim\pi_{\boldsymbol{\theta}}(\cdot|\boldsymbol{x})}\Big[\sum_{t=1}^{T}\Big(\sum_{t'=t}^{T}\big(\log\pi_{\boldsymbol{\theta}}(y_{t'}|\boldsymbol{x},\boldsymbol{y}_{<t'})-\log\pi^{*}(y_{t'}|\boldsymbol{x},\boldsymbol{y}_{<t'})\big)\Big)\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(y_t|\boldsymbol{x},\boldsymbol{y}_{<t})\Big]. \tag{17}$$

In practice, recent studies (Lu & Lab, 2025; Xiao et al., 2026) use a discount factor of 0 and approximate the gradient as

$$\nabla_{\boldsymbol{\theta}}\mathcal{J}_{\text{OPD}}(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{x}\sim D,\boldsymbol{y}\sim\pi_{\boldsymbol{\theta}}(\cdot|\boldsymbol{x})}\Big[\sum_{t=1}^{T}\big(\log\pi_{\boldsymbol{\theta}}(y_t|\boldsymbol{x},\boldsymbol{y}_{<t})-\log\pi^{*}(y_t|\boldsymbol{x},\boldsymbol{y}_{<t})\big)\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(y_t|\boldsymbol{x},\boldsymbol{y}_{<t})\Big]. \tag{18}$$

Similarly, the approximated gradient of G-OPD in Eq. (8) can be written as

$$\nabla_{\boldsymbol{\theta}}\mathcal{J}_{\text{G-OPD}}(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{x}\sim D,\boldsymbol{y}\sim\pi_{\boldsymbol{\theta}}(\cdot|\boldsymbol{x})}\Big[\sum_{t=1}^{T}A_t^{\text{G-OPD}}\,\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(y_t|\boldsymbol{x},\boldsymbol{y}_{<t})\Big], \tag{19}$$

where $A_t^{\text{G-OPD}} = \big(\log\pi_{\boldsymbol{\theta}}(y_t|\boldsymbol{x},\boldsymbol{y}_{<t})-\log\pi^{*}(y_t|\boldsymbol{x},\boldsymbol{y}_{<t})\big)+(\lambda-1)\big(\log\pi_{\text{ref}}(y_t|\boldsymbol{x},\boldsymbol{y}_{<t})-\log\pi^{*}(y_t|\boldsymbol{x},\boldsymbol{y}_{<t})\big)$.

B Detailed Training Settings

The training hyper-parameters in math and code RL training are put in Table 3 and Table 4 respectively. In RL, a reward of 1.0 is given when the final answer is correct in math reasoning or when all unit tests pass in code generation; otherwise, the reward is 0.0. Our experiments are based on `verl` (Sheng et al., 2024) framework. In both GRPO and G-OPD, we implement token-level rollout correction (Liu et al., 2025a) to mitigate training-inference mismatch.

The training hyper-parameters in G-OPD in both domains are in Table 5. In preliminary experiments, we find that under the same prompt size $*$ rollout n conditions, setting a larger prompt size leads to smoother convergence. The number of optimization steps for G-OPD in all experiments with same-size teacher-student pairs (Section 4.1) is set to 50, while it is set to 100 for experiments in the strong-to-weak distillation setting (Section 4.2). We find that further increasing the number of distillation steps may degrade generalization performance due to overfitting.

The training hyper-parameters in SFT are in Table 6. We make sure the number of trajectories to each problem generated by the teacher in SFT is consistent with that generated by the student in OPD and ExOPD. We keep the number of optimization steps consistent with the corresponding G-OPD experiment for fair comparison.

Table 3: Training hyper-parameters of GRPO in math RL.

Hyper-parameter	Value
Train Batch Size	128
Micro Batch Size	128
Rollout n	8
Maximum Prompt Length	2048
Maximum Response Length	16,384
Temperature	1.0
Top-p	1.0
LR	1×10^{-6}
Optimization Steps	500
KL Coefficient	0.0

Table 4: Training hyper-parameters of GRPO in code RL.

Hyper-parameter	Value
Train Batch Size	128
Micro Batch Size	128
Rollout n	8
Maximum Prompt Length	2048
Maximum Response Length	8192
Temperature	1.0
Top-p	1.0
LR	1×10^{-6}
Optimization Steps	300
KL Coefficient	0.0

Table 5: Training hyper-parameters of G-OPD in both math and code domains.

Hyper-parameter	Value
Batch Size	1024
Rollout n	1
Maximum Prompt Length	2048
Maximum Response Length	16,384
Temperature	1.0
Top-p	1.0
LR	1×10^{-5}

Table 6: Training hyper-parameters of SFT in both math and code domains.

Hyper-parameter	Value
Batch Size	1024
Maximum Sequence Length	32,768
Warm-up Ratio	0.05
LR	1×10^{-5}

C Additional Results under Single-Teacher On-Policy Distillation

Here, we display the additional results of reward interpolation and extrapolation variants of G-OPD under the experimental settings in Section 4.1.2 in Figure 3 and Figure 4.

D Comparison against Teachers with More RL Steps

We show the comparison against the teacher with more RL steps in Table 7 to demonstrate that the improvement of ExOPD over the teacher is not due to less training of the teacher, and more continued RL training on the teacher shows smaller improvement compared to ExOPD with fewer steps.

E Results of Distillation from Domain Teachers with Sufficient RL Trainings

Here, we show the on-policy distillation results when the domain teachers are trained with sufficient RL steps (i.e., 1200 steps). The experimental settings are the same as that in the Section 4.1. The results in Table 8 demonstrate the generalizability and effectiveness of ExOPD in this case.

F Training Dynamics of G-OPD

We display the training dynamics of ExOPD compared to OPD in the multi-teacher distillation setting in Figure 5.

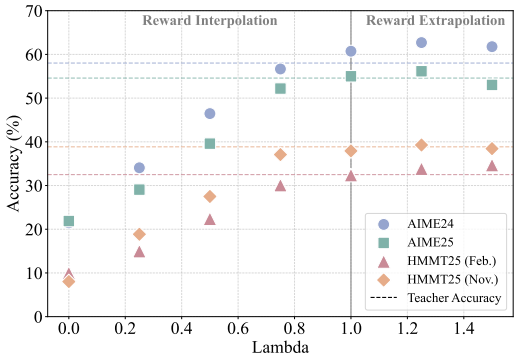


Figure 3: On-policy distillation results on four **math reasoning** benchmarks under different choices of reward scaling factor λ.

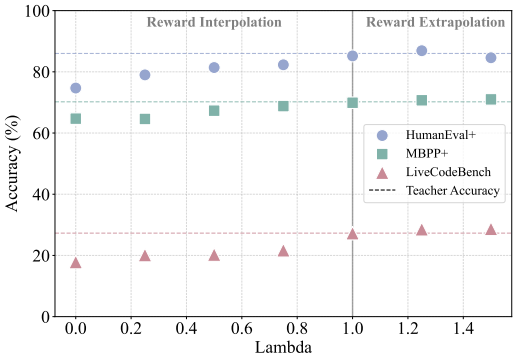


Figure 4: On-policy distillation results on three **code generation** benchmarks under different choices of reward scaling factor λ.

Table 7: Comparison against the math domain teacher with continued RL. Each numerical subscript indicates the absolute **improvement** or **degradation** **compared to the teacher model**.

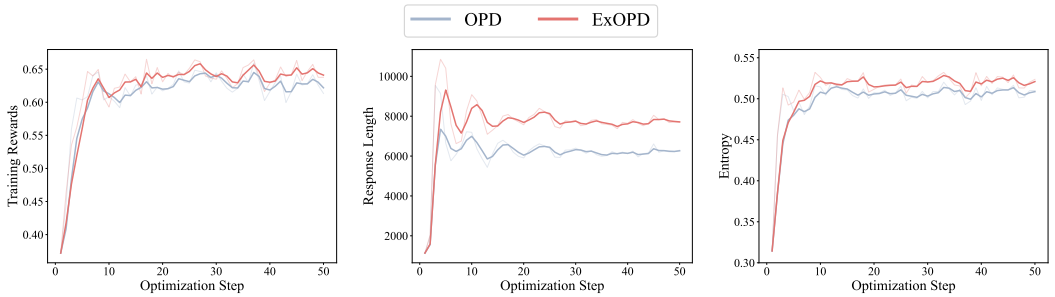
Method	AIME24	AIME25	HMMT25 (Feb.)	HMMT25 (Nov.)	Avg.
Teacher	58.0	54.6	32.5	38.9	46.0
+ continued RL (100 steps)	$60.9_{+2.9}$	$55.6_{+0.5}$	$32.8_{+0.3}$	$38.4_{-0.5}$	$46.9_{+0.9}$
ExOPD (50 steps)	$\mathbf{62.7}_{+4.7}$	$\mathbf{56.1}_{+1.5}$	$\mathbf{33.9}_{+1.4}$	$\mathbf{39.3}_{+0.4}$	$\mathbf{48.0}_{+2.0}$

G Prompt Templates

We show the prompt templates used in our experiments in the end.

Table 8: Distillation results under stronger teachers with sufficient RL trainings. Each numerical subscript indicates the absolute **improvement** or **degradation** **compared to the domain teacher model**.

Method	Math Reasoning					Code Generation			
	AIME24	AIME25	HMMT-25(Feb.)	HMMT-25(Nov.)	Avg.	Human-Eval+	MBPP+	LCB	Avg.
Teacher	68.2	59.3	37.3	42.9	51.9	88.9	72.5	28.0	63.1
Student	21.5	21.9	10.0	8.0	15.4	74.7	64.7	17.9	52.4
Single-Teacher Distillation									
OPD	$68.3_{+0.1}$	$58.7_{-0.6}$	$\mathbf{38.7}_{+1.4}$	$41.2_{-1.7}$	$51.7_{-0.2}$	$89.3_{+0.4}$	$71.3_{-1.2}$	$28.0_{+0.0}$	$62.9_{-0.2}$
ExOPD	$\mathbf{68.4}_{+0.2}$	$\mathbf{59.2}_{-0.1}$	$38.2_{+0.9}$	$\mathbf{42.8}_{-0.1}$	$\mathbf{52.2}_{+0.3}$	$\mathbf{89.9}_{+1.0}$	$\mathbf{73.7}_{+1.2}$	$\mathbf{29.3}_{+1.3}$	$\mathbf{64.3}_{+1.2}$
Multi-Teacher Distillation									
OPD	$68.2_{+0.0}$	$\mathbf{60.2}_{+0.9}$	$\mathbf{38.5}_{+1.2}$	$40.8_{-2.1}$	$51.9_{+0.0}$	$86.4_{-2.5}$	$72.1_{-0.4}$	$27.6_{-0.4}$	$62.0_{-1.1}$
ExOPD	$\mathbf{70.1}_{+1.9}$	$59.6_{+0.3}$	$37.5_{+0.2}$	$\mathbf{42.7}_{-0.2}$	$\mathbf{52.5}_{+0.6}$	$\mathbf{89.5}_{+0.6}$	$\mathbf{73.9}_{+1.4}$	$\mathbf{29.7}_{+1.7}$	$\mathbf{64.4}_{+1.3}$



(a) Training rewards (b) Response length (c) Entropy

Figure 5: Training dynamics of OPD and ExOPD in multi-teacher distillation experiments. We visualize using Exponential Moving Average (EMA) smoothing with a coefficient of 0.5.

Training and Evaluation Prompt Template for Math Reasoning

< |im_start| >user
{question}
Please reason step by step, and put your final answer within \boxed{}.< |im_end| >
< |im_start| >assistant

Training and Evaluation Prompt Template for Code Generation

< |im_start| >user
{question}
Write Python code to solve the problem. Present the code in
```python
Your code
```
at the end.
You need to think first then write the Python code.< |im_end| >
< |im_start| >assistant