ON-POLICY DELTA DISTILLATION

Anonymous authors
Paper under double-blind review

ABSTRACT

On-policy distillation is an alternative post-training method in reinforcement learning that alleviates the constraints imposed by reward models by providing token-level supervision from a teacher model. Although on-policy distillation has been studied and applied across various settings, its fundamental design remains underexplored. In this paper, we introduce a new distillation reward, termed *delta signal*, instead of directly imitating the teacher's output distribution. The *delta signal* is defined as the difference between the teacher model and its base model prior to instruction tuning for reasoning capability. It therefore captures the changes induced by reasoning tuning and provides a more direct signal for transferring reasoning capabilities. Using extensive empirical evidence, we show that the *delta signal* substantially improves on-policy distillation and refer to the new distillation method as On-Policy Delta Distillation (OPD^2). Experiments across mathematics, science, and code-reasoning benchmarks demonstrate that OPD^2 consistently outperforms conventional on-policy distillation, enabling reasoning LLMs to achieve strong performance with only a short post-training period. The code is included in the supplement and will be released publicly.

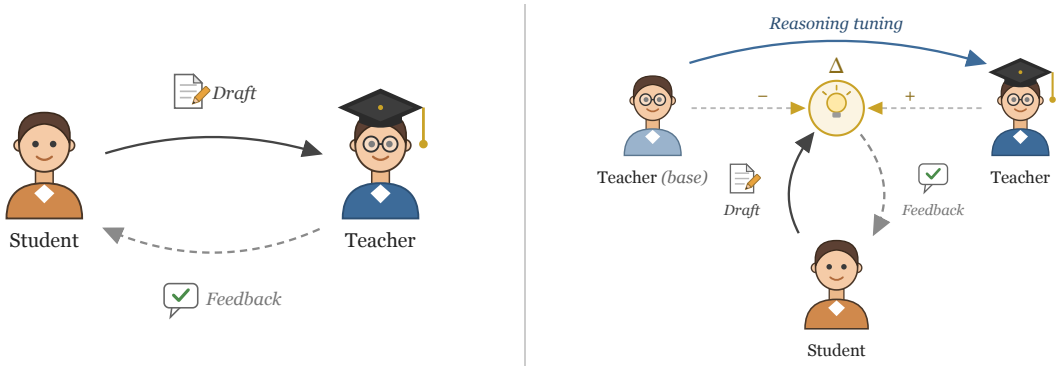
1 INTRODUCTION

As the use of large language models (LLMs) continues to expand, the demands placed on them have become increasingly diverse, driving continuous advances in training methods. While next-token prediction (Radford et al., 2018; Brown et al., 2020) on vast amounts of data remains a promising paradigm for pre-training, post-training processes have been extensively explored and remain key to aligning LLMs with downstream requirements. Post-training commonly involves supervised fine-tuning (SFT), which trains models on predefined desired outputs often generated by a larger and stronger LLM, and reinforcement learning (RL), which evaluates the quality of model-generated responses and optimizes the model toward more favorable outputs. Recently, On-Policy Distillation (OPD) has been actively studied as an alternative to reinforcement learning. With the benefits of large, high-performing models, OPD trains the model on dense signals from model-generated responses, reducing concerns about reward design and the noisy signals RL faces. Studies show that OPD can be competitive with RL in the post-training reasoning process, with reduced computational cost and improved performance, when a high-performing model is used as the teacher model.

On-Policy Distillation (OPD) is a variant of knowledge distillation (KD) similar to RL. Traditional KD (Hinton et al., 2015) is designed to transfer outputs from a high-performing teacher model to a student model. It is combined with the generative nature of LLMs and splits into two sub-categories. The first way is to distill from teacher-generated sequences. This approach was originally named Sequence-KD (Kim & Rush, 2016) and has been generalized to Supervised Fine-Tuning (SFT) on large model-generated sequences. It is widely used for the post-training stage just before RL. The second way is the on-policy distillation of student-generated sequences. For on-policy training, it generates sequences using the student network and uses the teachers to score the sampled tokens. The generated tokens that agree with the teacher are enhanced, while tokens with low probability on the teacher are degraded. As in RL, the learning signal is applied only to the sampled tokens. Thus, the student's original knowledge and ability are preserved throughout the post-training process. There are extensive studies (Li et al., 2025; Peng et al., 2025; Ma et al., 2026) that extend the OPD's learning signals beyond sampled tokens, but this paper will not cover these variants.

Despite its importance and prominence in post-training research, OPD's fundamental area remains underexplored. The rewards for OPD are simply set to log the difference in probability between the

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.



(a) On-policy distillation (OPD)

(b) On-Policy Delta Distillation (OPD²)

Figure 1: **Comparison of On-Policy Distillation (OPD) and our OPD^2.** Our OPD^2 utilizes the *delta signal*, the difference between the teacher and its base model, to focus on the knowledge of the learning trajectory required for reasoning capability.

teacher and the student. Even though OPD achieves remarkable performance with the original loss function, there may be room for improvement in the fundamental design of its loss function. Among the various directions for exploration, we focus on the learning trace of reasoning tuning (Mitchell et al., 2023). Most LLMs have multiple stages for training. It starts with pre-training on next-token prediction tasks using massive amounts of data. Then, tuned for reasoning and chat ability with post-training such as SFT and RL. As a post-training method, OPD only requires knowledge from the post-training phase. Thus, as shown in Fig. 1, we design a new loss function for on-policy distillation based on the difference between the base and the reasoning tuning model of the teacher network.

Our core concept is described in Fig. 1. In OPD, the teacher model acquires reasoning ability through reasoning tuning (*e.g.,* instruction tuning). However, the reasoning-tuned teacher also retains natural preferences and stylistic tendencies acquired before the reasoning tuning. We argue that distilling the reasoning-tuned teacher alone does not necessarily reveal the learning trajectory that led the teacher to acquire reasoning capabilities. Thus, we can provide reasoning knowledge by utilizing the delta between the reasoning-tuned teacher and the base teacher (*i.e.,* the model before post-training). We propose On-Policy Delta Distillation (OPD^2), defining the *delta signal* using it as the main reward for OPD. In contrast to OPD, which uses the log probability difference between teacher and student as the reward signal, OPD^2 uses the difference between teacher and teacher base as the reward signal. We first conduct three analyses to show the differences and benefits of the *delta signal* relative to the original reward: word clouds for distillation signals, token-level visualizations, and word-level statistical analyses. These analyses highlight the differences and characteristics of Δ, providing insight into changes in reward. Based on these analyses, we find that the *delta signal* offers an effective alternative distillation signal with distinct and desirable properties.

Building on this observation, we introduce two reward designs: centering and joint conditioning, to effectively utilize the *delta signal* for OPD. First, centering subtracts the expected rewards from the policy model's sampling probabilities to make the signal easier to handle. In the original OPD, it is automatically removed through training on-policy probabilities. But it is hard to determine the direction of the actual learning signal, and it can cause negative effects without a single sign-biased condition. Using zero-centered rewards, we introduce a joint condition between OPD and the *delta signal* to address the convergence-point issue. With these two design points, we complete the reward design for our OPD^2 and successfully introduce the *delta signal* into the OPD framework.

In our experiments, we develop an extensive verification framework across three reasoning domains: Math, Science, and Code. We construct a mixed-domain training set by sampling an equal number of questions from one dataset for each of the Math (Moshkov et al., 2025), Science (NVIDIA Corporation, 2025), and Code (Ahmad et al., 2025) domains, resulting in a 1:1:1 domain ratio. We then perform on-policy distillation on this mixed training set. The models trained by on-policy distillation methods are evaluated on 7 Math, 3 Science, and 4 Code benchmarks. Our verification covers various sizes of Qwen3 (Yang et al., 2025), both non-thinking and thinking modes, and the recently released Gemma-4 (Gemma Team, 2026). In extensive verification, OPD^2 consistently

outperforms other on-policy distillation methods, demonstrating an impressive performance gap. It is noteworthy that OPD^2 significantly improves on-policy distillation across various sizes (1.7B - 8B), both in no-think and think modes, on cutting-edge models with superior reasoning performance (Yang et al., 2025; Gemma Team, 2026).

2 METHOD

2.1 PRELIMINARY

On-Policy Distillation (OPD) is a post-training method that can replace reinforcement learning methods such as GRPO by leveraging efficient per-sample computation with token-level supervision. Qwen3 highlights the practical efficiency of OPD on its small models (Yang et al., 2025). Subsequent studies (Xiao et al., 2026; Yang et al., 2026) have developed detailed OPD formulations and implementations. Basically, knowledge distillation trains the student to reduce the output probability distance from the teacher. Since the output is a probability, Cross-Entropy or Kullback-Leibler divergence is employed for the distance function. The loss function of OPD is defined as knowledge distillation loss on on-policy data, *i.e.,* the student's rollout samples as

$$\mathcal{L}_{\mathrm{OPD}}(\theta) = \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_\theta(\cdot|x)} \left[D_{\mathrm{KL}} \left(\pi_\theta(y \mid x) \,\|\, \pi^*(y \mid x) \right) \right], \tag{1}$$

where π_θ indicates the student and π^* is the teacher model. For a given question x, OPD generates a student response $y \sim \pi_\theta(\cdot \mid x)$ and trains the student's next-token prediction to mimic that of the teacher by minimizing KL divergence.

OPD is often used to replace reinforcement learning in the post-training stage. Thus, OPD implementation also follows an RL-like framework to leverage optimized on-policy training codes and settings. Similar to the RL setting, the OPD objective is defined as a reward applied only to sampled tokens. For the question x and given context $y_{<t}$, the reward for t-th token y_t is defined as

$$R_t^{\mathrm{OPD}} = \log \pi^*(y_t \mid x, y_{<t}) - \log \pi_\theta(y_t \mid x, y_{<t}). \tag{2}$$

Note that the reward R_t is only applied to determine whether to enhance or degrade the sampled token y_t. It doesn't make gradients for unsampled probabilities. While there are studies on top-k OPD that address unsampled tokens for both rewards and training, we will not cover them in the paper. Eq. 2 is obtained from the partial derivative of the KL divergence with respect to the sampled-token probability $\pi_\theta(y_t \mid x, y_{<t})$. Since the reward is maximized whereas the KL divergence is minimized, we reverse the sign and omit the additive constant -1.

Using R_t as token-level rewards, OPD trains the student with the following gradient computation:

$$\nabla_\theta J_{\mathrm{OPD}}(\theta) = \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_\theta(\cdot|x)} \left[\sum_{t=1}^{T} R_t^{\mathrm{OPD}} \nabla_\theta \log \pi_\theta(y_t \mid x, y_{<t}) \right]. \tag{3}$$

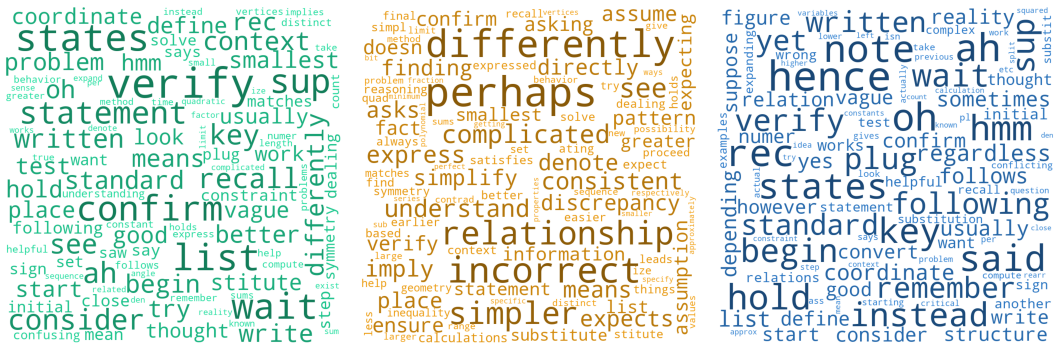
With this implementation, the student network is trained to maximize the token-level rewards R_t, which is equivalent to minimizing the KL divergence on on-policy data.

2.2 DELTA SIGNAL FOR DISTILLATION

OPD uses the log-probability difference on the sampled logit y_t as rewards for the token as in Eq. 2, where π^* presents the teacher model and π_θ means the student model. Following the teacher is fundamental to knowledge distillation. But we propose to go one step further by using the teacher's base model (*i.e.,* the model prior to post-training), π^*_{base}. In other words, we define the *delta signal*:

$$R_t^{\Delta} = \log \pi^*(y_t \mid x, y_{<t}) - \log \pi^*_{base}(y_t \mid x, y_{<t}) \tag{4}$$

and use it as a primary learning objective for on-policy distillation. R_t^{Δ} represents the learning trace of the teacher model from its own base model. In general, LLMs are pre-trained on next-token prediction on massive data, then tuned with SFT and RL to improve their reasoning ability. As shown



(a) OPD ($\log\pi^*$ - $\log\pi_\theta$) (b) Base ($\log\pi^*_{base}$ - $\log\pi_\theta$) (c) Δ ($\log\pi^*$ - $\log\pi^*_{base}$)

Figure 2: **Word clouds for OPD, Base and Δ.** The figure illustrates signal strengths as word clouds. Qwen3-1.7B and -4B are used for the student and teacher, respectively, on 10k math questions. Compared with OPD, Δ emphasizes reasoning-connective words such as *hence*, *however*, and *instead*, while suppressing exploratory and verification-related words such as *see*, *try*, and *verify*.

in Fig 1, the target knowledge for on-policy distillation is reasoning ability, not basic next-token prediction. Thus, in our on-policy distillation, R_t^Δ serves as the primary learning signal, extracting the teacher's knowledge to enable complex reasoning beyond simple next-token prediction.

Before applying R^Δ for on-policy distillation, we analyze the difference between R^{OPD} and R^Δ to give intuition behind this change. We provide three analyses of R^Δ: word clouds to show signal patterns, signal differences with a simple question, and statistical signal changes from R^{OPD} to R^Δ.

Word clouds. Fig. 2 shows the signal pattern for OPD, Base, and Δ. OPD represents words enhanced by conventional on-policy distillation loss. These words are preferred by the teacher model among the sampling candidates in the student policy. Base signal means the signals preferred by the base model, which are not connected with reasoning ability and are described as a negative signal in Fig. 1. We conjecture that these natural preferences are also included in the teacher model and don't need to be transferred to the student for reasoning performance. The last Δ presents words enhanced when distillation is conducted with Eq. 4. Compared to OPD, Δ enhances reasoning connection words such as *hence*, *note*, *instead*, *however*, and *yet*. On the other hand, words that verify and explore the solution are degraded compared to OPD: *see*, *try*, *verify*, and *confirm*. As expected, the degraded words also appear in the base model's word cloud.

Token-level signals. Fig. 3 shows the token-level reward signals for OPD and Δ. OPD and Δ show a meaningful difference in various tokens of reasoning. In general, Δ is more negative than OPD on generic reasoning expressions such as *There*, *Since*, and *The variable*. In the first question, the reasoning goes wrong at *multiply*. At those parts, Δ shows relatively negative signals compared to OPD, including the "$\times$" token. It is similar to other questions. In the second, *into steam* is positive on OPD while negative on Δ. The critical reasoning parts in the third question are *joins* and *values together*, which also show different signal directions for OPD and Δ. OPD keeps showing positive signals in the incorrect reasoning part. Both the teacher and student show strong negative signals in that part. But the student signals are stronger than the teacher's. As a result, OPD produces positive signals even in traces of wrong reasoning. In the case of Δ, two models originate from the same weights, and the reasoning-tuned teacher is more sensitive to incorrect tokens. Thus, the teacher and base model have a superior-inferior relationship in reasoning about related words, which makes Δ more robust than OPD in the correlation between the distillation signal and reasoning correctness.

2.3 On-policy Delta Distillation (OPD^2)

Although the *delta signal* (R^Δ) can be beneficial for distillation, it may have an issue with its convergence behavior. Specifically, unlike the original reward, R^Δ does not consider the student signal π_θ in the reward computation; as a result, the training based on R^Δ converges to the maximum reward token with a one-hot vector. Even though this convergence point is not reachable in a practical

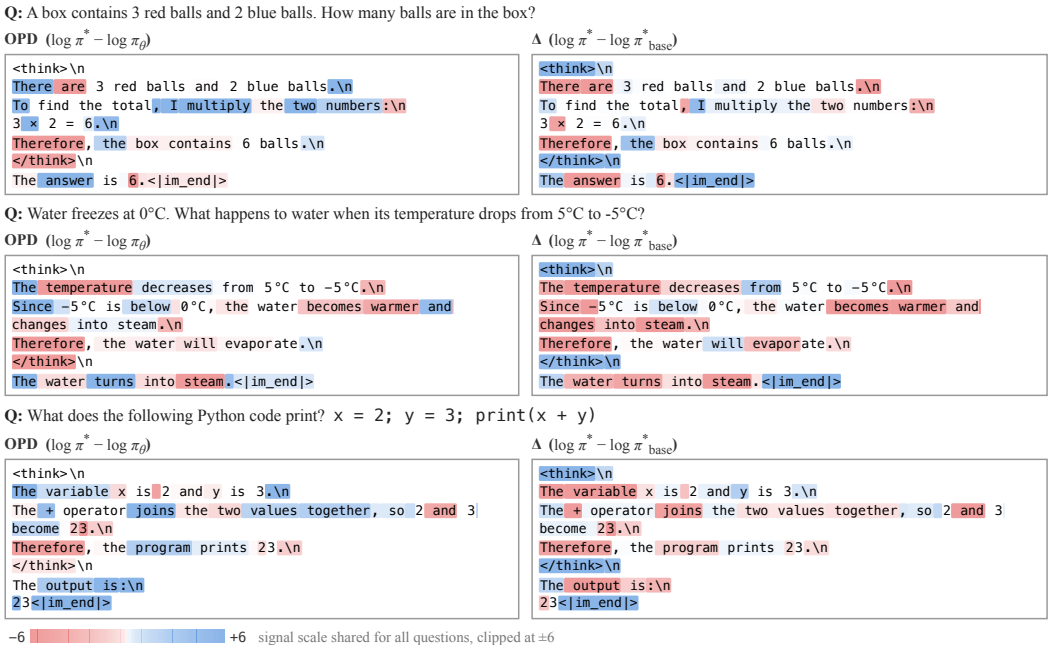


Figure 3: **Token-level distillation signals.** Blue and red indicate promoting and suppressing signals, respectively. Δ more consistently suppresses tokens aligned with incorrect reasoning than OPD.

strong-to-weak setting, it might still cause instability during training. Thus, we design a convergence point for R^{Δ} such that no gradients are computed when the student models match the teacher.

On-policy distillation shares the on-policy nature with RL. Thus, a bias invariant to the action (*i.e.,* token selection) in the rewards is ineffective for training (Williams, 1992; Sutton et al., 1999; Mnih et al., 2016). We therefore begin with the bias-subtracted OPD formulations, which obtain the advantage by subtracting the expected reward under the on-policy distribution:

$$A_t^{\text{OPD}} = R_t^{\text{OPD}} - \mathbb{E}_{\tilde{y}_t \sim \pi_\theta(\cdot|x, y_{<t})} \left[\log \pi^*(\tilde{y}_t \mid x, y_{<t}) - \log \pi_\theta(\tilde{y}_t \mid x, y_{<t}) \right], \tag{5}$$

$$A_t^{\Delta} = R_t^{\Delta} - \mathbb{E}_{\tilde{y}_t \sim \pi_\theta(\cdot|x, y_{<t})} \left[\log \pi^*(\tilde{y}_t \mid x, y_{<t}) - \log \pi^*_{base}(\tilde{y}_t \mid x, y_{<t}) \right]. \tag{6}$$

A_t^{OPD} and A_t^{Δ} represent reward advantages of sampled action y_t over expected rewards values over its original probability $\pi_\theta(\cdot \mid x, y_{<t})$. In practice, we compute the expected reward over the top-k (k=1024) tokens to save GPU memory. We use A_t^{Δ} to replace R_t^{OPD} for the on-policy distillation driven by Δ signals. For the convergence issue, we add a stop condition in alignment with A_t^{OPD} as

$$A_t^{D^2} = \begin{cases} A_t^{\Delta} & \text{if } A_t^{\Delta} A_t^{\text{OPD}} > 0, \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

$A_t^{D^2}$ is our advantage function for OPD^2. It basically follows A_t^{Δ} but is turned off when it conflicts with original distillation signals. The condition ($A_t^{\Delta} A_t^{\text{OPD}} > 0$) prevents the student from being over-trained by the trace signal and being distant from the teacher's signal. When the student probability is the same as that of the teacher, *i.e.,* $\pi_\theta = \pi^*$, the advantage $A_t^{D^2}$ is turned off by the condition. On-policy distillation with $A_t^{D^2}$ restricts updates to sign-consistent common descent directions between the trace A_t^{Δ} and distillation A_t^{OPD} signals, while using A_t^{Δ} to control the gradient magnitude. Thus, in RL form, the training gradient is described as

$$\nabla_\theta J_{\text{OPD}^2}(\theta) = \mathbb{E}_{x \sim \mathcal{D},\, y \sim \pi_\theta(\cdot|x)} \left[\sum_{t=1}^{T} A_t^{D^2} \nabla_\theta \log \pi_\theta(y_t \mid x, y_{<t}) \right]. \tag{8}$$

3 EXPERIMENT

We design an extensive evaluation framework to verify the performance improvements of our OPD^2. Since on-policy distillation is primarily used as a post-training process, we target small reasoning models, such as Qwen3 (Yang et al., 2025) and Gemma4 (Gemma Team, 2026), as student models, using large models from the same family as the teacher. As a post-training process, a notable benefit of on-policy distillation is that it can be applied to any domain without requiring domain-specific reward design. To elaborate on this advantage, we build a multi-domain training dataset by mixing sampled questions from three reasoning domains: Math, Science, and Code. After on-policy distillation, the trained model is evaluated on various benchmarks, including 7 Math, 3 Science, and 4 Code benchmarks. Note that our benchmark covers both non-thinking and thinking modes for the Qwen3 architecture. We also report training dynamics, an ablation study, and computational analysis for OPD^2, which help to understand detailed characteristics of OPD^2 training. Note that the detailed experiment settings are included in the appendix: Tables 10 and 11.

3.1 EXPERIMENT SETTINGS

Models. We target open-source models with strong reasoning abilities: Qwen3 (Yang et al., 2025) and Gemma4 (Gemma Team, 2026). Note that all student and teacher models are instruct-tuned, while the teacher-base model is -Base. Recently, on-policy distillation studies have reported non-thinking mode performance rather than thinking-mode performance for Qwen3. We find that both settings offer distinct and complementary value. Qwen3 is not well-tuned for reasoning tasks in non-thinking mode. Thus, there is enough gap for improvement, and on-policy distillation has to make substantial changes on student models to achieve comparable performance. On the other hand, thinking mode already has top-level reasoning ability. Thus, on-policy distillation faces the challenges of improving or altering the student model without compromising its existing performance. Therefore, we evaluate both modes for Qwen3, whereas only the thinking mode is verified for Gemma4.

Training data. We introduce a multi-domain post-training scenario for on-policy distillation. Three datasets are selected for three reasoning domains: OpenMathReasoning (Moshkov et al., 2025), OpenScienceReasoning-2 (NVIDIA Corporation, 2025), and OpenCodeReasoning (Ahmad et al., 2025). These datasets contain challenging questions, which are enough to extract meaningful distillation signals with on-policy distillation. We conduct 1:1:1 balanced sampling for each domain and make a subset with 100k questions for on-policy distillation. Since the training uses fewer than 30k samples, the model experiences only a fraction of the available questions, corresponding to less than one training epoch. Each question is used at most once. Note that we use only questions from these datasets. i.e., the reasoning traces and answers are discarded from the data during sampling.

Evaluation. The models trained with the on-policy distillation methods are evaluated on various benchmarks. We select several benchmarks that are available in Evalchemy (Guha et al., 2024) or lm-eval-harness (Gao et al., 2024). 7 Math, 3 Science, and 4 Code benchmarks are selected as the evaluation set. For math, the models are evaluated on AIME24 (MAA, 2024), AIME25 (MAA, 2025), AMC23 (MAA), HMMT25 (Dekoninck et al., 2026), MATH500 (Hendrycks et al., 2021), OlympiadBench (He et al., 2024), and ReasoningGym Math (Stojanovski et al., 2025). For coding, we use CodeContests (Li et al., 2022), CodeForces (Guha et al., 2025), LiveCodeBench (Jain et al., 2024), and ReasoningGym Algorithm (Stojanovski et al., 2025). GPQA (Rein et al., 2024), SuperGPQA (Team et al., 2025), and SciBench (Wang et al., 2024) are selected for science. All benchmarks use the pass@1 metric with averages across repetitions as listed in Table 12.

3.2 REASONING PERFORMANCE

Based on the extensive evaluation, we compare the performance of OPD^2 with the original model, OPD, and ExOPD (Yang et al., 2026). Since the original models are reasoning-tuned and have strong reasoning abilities, on-policy distillation methods are first evaluated to determine whether they are beneficial to the model. OPD means basic on-policy distillation training with Eq. 3. Although it is the most fundamental on-policy distillation method, it still shows promising performance improvements and is applied in various practical settings (Yang et al., 2025; Xiao et al., 2026). ExOPD (Yang et al., 2026) is a recently proposed advanced method for on-policy distillation. It leverages a teacher–base

Table 1: **Qwen3 non-thinking results.** The table shows non-thinking mode results for Qwen3 models. The best performance is shown in bold, and the second best is underlined.

	Math							
Model	AIME24	AIME25	AMC23	HMMT25	MATH500	Olympiad	RGMath	**Avg**
Qwen3-1.7B	14.2	9.4	41.2	5.0	68.6	26.0	79.5	34.8
+ OPD	<u>36.7</u>	<u>23.8</u>	70.8	**17.0**	81.0	37.1	<u>90.9</u>	51.0
+ ExOPD	35.4	23.3	<u>75.5</u>	14.7	<u>81.9</u>	<u>37.8</u>	<u>90.9</u>	<u>51.4</u>
+ OPD^2	**41.0**	**28.8**	**79.5**	<u>15.0</u>	**83.9**	**40.1**	**93.9**	**54.6**
Qwen3-4B	21.9	18.5	63.8	12.3	79.8	33.4	90.6	45.8
+ OPD	53.3	46.9	89.5	31.3	89.5	43.8	<u>91.8</u>	63.7
+ ExOPD	<u>63.5</u>	<u>48.5</u>	<u>91.2</u>	<u>34.3</u>	<u>90.1</u>	<u>44.4</u>	91.6	<u>66.3</u>
+ OPD^2	**71.5**	**59.0**	**94.8**	**38.0**	**91.4**	**46.5**	**95.2**	**70.9**

	Code					Science			
Model	Code Contests	Code Forces	LCBv5	RG Algo	**Avg**	GPQA	Super GPQA	Sci Bench	**Avg**
Qwen3-1.7B	4.8	10.9	7.0	19.4	10.6	39.4	26.1	26.9	30.8
+ OPD	10.3	16.6	27.9	30.0	21.2	<u>48.4</u>	26.1	35.0	36.5
+ ExOPD	<u>17.0</u>	<u>19.7</u>	<u>29.9</u>	<u>33.6</u>	<u>25.0</u>	48.0	<u>26.2</u>	<u>35.4</u>	<u>36.5</u>
+ OPD^2	**26.7**	**23.4**	**35.0**	**37.7**	**30.7**	**49.8**	**27.3**	**39.2**	**38.8**
Qwen3-4B	12.9	18.2	24.0	33.6	22.2	42.3	32.6	45.2	40.0
+ OPD	17.6	26.4	40.8	46.0	32.7	54.9	<u>34.4</u>	52.2	47.2
+ ExOPD	<u>25.5</u>	<u>30.3</u>	<u>45.2</u>	<u>52.5</u>	<u>38.4</u>	<u>55.2</u>	34.0	<u>54.2</u>	<u>47.8</u>
+ OPD^2	**36.6**	**32.7**	**47.4**	**63.4**	**45.0**	**57.5**	**36.0**	**55.7**	**49.7**

model to amplify the extrapolation signal and represents a state-of-the-art approach in this area. We implement ExOPD with $\lambda = 1.25$, following the setting reported in the original paper.

Non-thinking mode. We evaluate all methods in the non-thinking mode, where the models generate answers without explicit reasoning traces. As shown in Table 1 and 6, all on-policy distillation methods substantially improve the original models across Math, Code, and Science, suggesting that on-policy distillation is effective at enhancing the non-thinking capabilities of reasoning models. Among the methods compared, OPD^2 achieves the best average performance across all model sizes and domains. The improvements are particularly notable in Math. For Qwen3-1.7B, OPD^2 increases the average score from 34.8 to 54.6, outperforming OPD and ExOPD. Similar gains are observed for Qwen3-4B and Qwen3-8B. OPD^2 also consistently improves performance in Code and Science. On Code, the gain is especially large for Qwen3-1.7B, improving the original model by 20.1 points and ExOPD by 5.6 points. On Science, OPD^2 obtains the highest average score for all three model sizes. These results indicate that the advantages of OPD^2 are not restricted to mathematical reasoning but generalize across diverse reasoning domains.

Thinking mode. We also evaluate the methods in the thinking mode, where the Qwen3 models exhibit strong reasoning ability with explicit reasoning traces. Table 2 shows thinking mode reasoning performance compared to non-thinking mode. Note that the full table is also included in the appendix, Tables 7 and 8. Improving thinking mode performance is substantially more challenging than improving their relatively weak non-thinking capabilities. In particular, standard OPD often degrades performance, while ExOPD provides only limited improvements. For example, on Math, both methods underperform the original models across all three model sizes on average. In contrast, OPD^2 consistently achieves the highest performance across all model sizes and domains. On Math, OPD^2 improves the original 1.7B and 8B models by 3.5 and 2.2 points, respectively. A similar trend is observed in Code and Science. On Code, OPD^2 improves the average scores of the 1.7B and 8B models from 29.5 and 51.2 to 40.7 and 59.6, respectively. In particular, the 8B model obtains consistent gains across all four code benchmarks. On Science, OPD^2 still achieves the best average performance for every model size, whereas OPD and ExOPD often reduce the baseline performance. These results demonstrate that OPD^2 can improve not only weak non-thinking capabilities but also the strong reasoning capabilities in thinking mode.

Table 2: **Qwen3 thinking and non-thinking modes.** The table reports average performance across Math, Code, and Science benchmarks in non-thinking and thinking modes.

	Qwen3-1.7B						Qwen3-8B					
	Non-thinking			Thinking			Non-thinking			Thinking		
Model	Math	Code	Sci.	Math	Code	Sci.	Math	Code	Sci.	Math	Code	Sci.
Original	34.8	10.6	30.8	59.2	29.5	40.1	46.9	25.0	43.5	73.7	51.2	53.0
+ OPD	51.0	21.2	36.5	57.1	32.6	41.7	65.9	35.3	49.7	72.3	52.4	50.9
+ ExOPD	51.4	25.0	36.5	58.4	37.2	42.2	67.8	42.3	50.2	73.6	55.9	52.0
+ OPD^2	**54.6**	**30.7**	**38.8**	**62.7**	**40.7**	**43.5**	**71.6**	**44.6**	**51.6**	**75.9**	**59.6**	**54.6**

Table 3: **Gemma-4.** The table shows on-policy distillation results for Gemma4-E4B-it. The best performance is shown in bold, and the second best is underlined.

	Math							
Model	AIME24	AIME25	AMC23	HMMT25	MATH500	Olympiad	RGMath	**Avg**
Gemma4-E4B-it	51.7	37.9	88.8	27.3	87.2	41.6	89.8	60.6
+ OPD	49.4	35.6	84.5	25.0	82.9	41.5	93.4	58.9
+ ExOPD	59.6	41.7	90.0	38.0	87.8	**46.0**	94.2	65.3
+ OPD^2	**69.2**	**44.0**	**92.3**	**40.0**	**88.3**	45.5	**95.7**	**67.8**

	Code					Science			
Model	Code Contests	Code Forces	LCBv5	RG Algo	Avg	GPQA	Super GPQA	Sci Bench	Avg
Gemma4-E4B-it	**52.1**	**43.1**	**63.7**	62.7	**55.4**	**59.9**	34.4	46.8	47.0
+ OPD	12.3	26.0	47.7	60.4	36.6	46.0	32.9	38.1	39.0
+ ExOPD	29.5	35.1	53.8	63.7	45.5	55.4	36.4	50.2	47.3
+ OPD^2	44.4	36.8	50.6	**64.0**	49.0	57.8	**37.7**	**50.9**	**48.8**

Gemma4. Additionally, we evaluate the methods on Gemma4-E4B-it (Gemma Team, 2026) to examine whether their effectiveness generalizes beyond the Qwen3 model family. As shown in Table 3, standard OPD substantially degrades performance across all three domains, indicating that directly applying conventional on-policy distillation can degrade performance when transferred to a different model family. ExOPD alleviates this degradation and yields meaningful improvements in Math, but its effectiveness remains inconsistent across domains.

OPD^2 achieves the best Math performance by a clear margin, improving the average score from 60.6 to 67.8. It outperforms ExOPD by 2.5 points on average and achieves the highest score on six of the seven benchmarks. The results on Code present a more challenging case. Gemma4-E4B-it already exhibits strong coding performance, and none of the distillation methods surpasses the original model on average. Nevertheless, OPD^2 retains substantially more of the original capability than OPD and ExOPD. OPD^2 also improves RGAlgo from 62.7 to 64.0. On Science, OPD^2 improves the average score from 47.0 to 48.8 and achieves the best results on SuperGPQA and SciBench. Overall, these results show that OPD^2 generalizes across model families and provides a more robust trade-off between improving target capabilities and preserving the original model's strong capabilities.

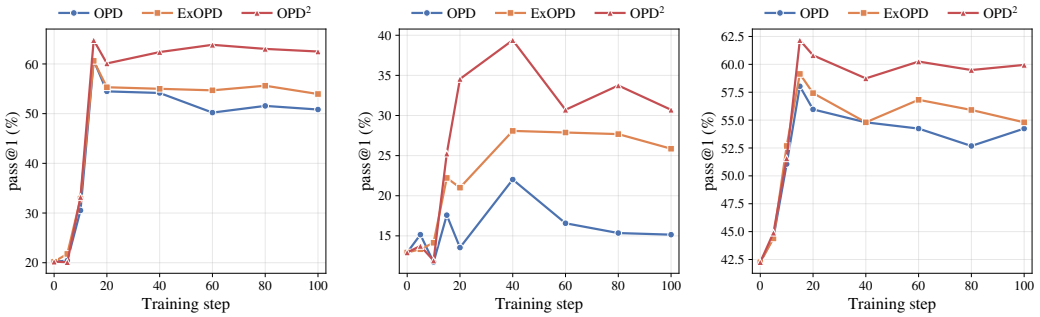
3.3 Training dynamics

OPD^2 demonstrates substantial performance improvements across all domains and consistently outperforms the other on-policy distillations with meaningful margins. To provide additional information for training, we compare the learning trajectories of OPD, ExOPD, and OPD^2 under the Qwen3-4B non-thinking setting in Fig. 4. All methods show a rapid performance increase in early training stage, indicating that most benefits of on-policy distillation are achieved within a relatively small number of optimization steps. After this initial improvement, however, their trajectories diverge.

OPD and ExOPD tend to peak early and then plateau or degrade as training proceeds. This behavior is particularly evident on Math and Code, where the final performance can be noticeably lower than the intermediate peak. In contrast, OPD^2 achieves a larger initial improvement and maintains a consistently higher performance throughout the remaining training steps. Although its performance

Table 4: **Ablation study for OPD^2.** Replacing the *delta signal* shows the most significant impact. The formulas indicate how each variant modifies Eq. 7.

	Non-thinking			Thinking		
	Math	Code	Science	Math	Code	Science
OPD^2	54.6	30.7	38.8	62.7	40.7	43.5
No *delta signal* ($A_t^{\Delta} \to A_t^{\mathrm{OPD}}$)	50.5	22.8	35.9	57.4	32.5	41.3
No condition ($\cancel{A_t^{\Delta} A_t^{\mathrm{OPD}} > 0}$)	55.8	30.7	38.8	61.5	39.3	43.8
No centering ($A_t^{\Delta} \to R_t^{\Delta}$)	54.9	30.6	38.8	63.1	39.7	43.3



(a) AIME24 & 25 (b) CodeContests (c) GPQA

Figure 4: **Training curve.** The figures show changes in reasoning performance during on-policy distillation. We report the average performance on AIME24 and AIME25, CodeContests, and GPQA. All OPD variants show similar dynamics, while OPD^2 achieves the best performance.

also fluctuates after the early peak, it remains clearly above that of OPD and ExOPD across all three domains. Please note that all results in the main tables are reported at the final training step. We do not select the best-performing checkpoint.

3.4 Ablation Study

We conduct an ablation study to analyze each component's contribution in OPD^2. As shown in Table 4, replacing the *delta signal* with the standard OPD signal causes the largest performance degradation in both non-thinking and thinking modes. This result indicates that the *delta signal* is the primary source of the performance gains achieved by OPD^2. Removing the agreement condition or the centering operation has a comparatively smaller effect. Although these components generally contribute to the overall performance, their impact is not as substantial or consistent as that of the *delta signal*. Overall, the ablation results show that the *delta signal* plays the central role in OPD^2, while the condition and centering provide additional but relatively modest improvements.

4 Conclusion

In this paper, we have examined the potential of the *delta signal*, the difference between the teacher and the base model, as the major training signal for on-policy distillation. We conjectured that the *delta signal* can be an effective way to transfer reasoning knowledge obtained through reasoning tuning of the teacher model, starting from base models. Through various analyses, we showed that the *delta signal* includes meaningful reasoning signals and has beneficial points compared to the original distillation rewards. Based on this observation, we have proposed OPD^2, on-policy distillation based on the *delta signal*. Our OPD^2 is verified on an extensive framework comprising three reasoning domains (Math, Science, and Code), 14 evaluation benchmarks, and networks of various sizes (1.7B-8B) across cutting-edge models (Qwen3 and Gemma4). In all settings, OPD^2 achieved substantial performance improvements over conventional on-policy distillation, demonstrating the effectiveness of the *delta signal* for on-policy distillation. We believe that OPD^2 can serve as an important milestone in the development of on-policy distillation.

AI USE STATEMENT

In this work, we used generative AI tools to provide feedback on research methodology and experiments, reformat datasets, assist with method implementation, support qualitative and thematic data analysis, and interpret results. We have not used generative AI tools to help develop conceptual frameworks, assist with translation, propose or refine hypotheses, or to design research methodology or experiments. Generating synthetic data sets, helping develop theoretical models, formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, and cleaning datasets are not applicable to this work. Additionally, we used generative AI tools for creating and modifying scientific figures, creating artifacts, drafting parts of a research paper, summarizing and analyzing existing literature, brainstorming, editing a research paper to improve readability. We have reviewed all AI-assisted work. LLM-generated code, data, analyses, figures, and other artifacts were reviewed and verified for correctness by the authors. The authors developed and determined the original research ideas, core concepts, methodological decisions, experimental designs, and scientific conclusions. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

To improve reproducibility, we provide detailed implementation details mainly in Section C. The detailed training settings are denoted in Table 11 for common parameters and Table 10 for the model-specific parameters. The evaluation benchmark repetition numbers are included in Table 12. We also include our source code in the supplementary material, including training code for all model variants (Qwen3-1.7B, -4B, -8B, and Gemma-4-E4B) and evaluation code for all 14 benchmarks. We verified that the code reproduces our main table results with a small statistical gap. We will release the code publicly to help researchers in the on-policy distillation field reproduce our results.

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 *International Conference on Learning Representations*, volume 2024, pp. 21246–21263, 2024.

Wasi Uddin Ahmad, Sean Narenthiran, Somshubra Majumdar, Aleksander Ficek, Siddhartha Jain, Jocelyn Huang, Vahid Noroozi, and Boris Ginsburg. Opencodereasoning: Advancing data distillation for competitive coding. *arXiv preprint arXiv:2504.01943*, 2025.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. *Advances in neural information processing systems*, 28, 2015.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Jasper Dekoninck, Nikola Jovanović, Tim Gehrunger, Kári Rögnvaldsson, Ivo Petrov, Chenhao Sun, and Martin Vechev. Beyond benchmarks: Matharena as an evaluation platform for mathematics with llms. 2026. URL https://arxiv.org/abs/2605.00674.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024. URL https://zenodo.org/records/12608602.

Gemma Team. Gemma 4 technical report. *arXiv preprint arXiv:2607.02770*, 2026. doi: 10.48550/arXiv.2607.02770.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Minillm: On-policy distillation of large language models, 2026. URL https://arxiv.org/abs/2306.08543.

Etash Guha, Negin Raoff, Jean Mercat, Ryan Marten, Eric Frankel, Sedrick Keh, Sachin Grover, George Smyrnis, Trung Vu, Jon Saad-Falcon, Caroline Choi, Kushal Arora, Mike Merrill, Yichuan Deng, Ashima Suvarna, Hritik Bansal, Marianna Nezhurina, Reinhard Heckel, Seewong Oh, Tatsunori Hashimoto, Jenia Jitsev, Yejin Choi, Vaishaal Shankar, Alex Dimakis, Mahesh Sathiamoorthy, and Ludwig Schmidt. Evalchemy, November 2024.

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.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems, 2024. URL https://arxiv.org/abs/2402.14008.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset, 2021. URL https://arxiv.org/abs/2103.03874.

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

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.

Woogyeol Jin, Taywon Min, Yongjin Yang, Swanand Ravindra Kadhe, Yi Zhou, Dennis Wei, Nathalie Baracaldo, and Kimin Lee. Entropy-aware on-policy distillation of language models. *arXiv preprint arXiv:2603.07079*, 2026.

Jeonghye Kim, Xufang Luo, Minbeom Kim, Sangmook Lee, Dohyung Kim, Jiwon Jeon, Dongsheng Li, and Yuqing Yang. Why does self-distillation (sometimes) degrade the reasoning capability of llms? *arXiv preprint arXiv:2603.24472*, 2026.

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.

Jongwoo Ko, Sungnyun Kim, Tianyi Chen, and Se-Young Yun. Distillm: Towards streamlined distillation for large language models. *arXiv preprint arXiv:2402.03898*, 2024.

Jongwoo Ko, Sara Abdali, Young Jin Kim, Tianyi Chen, and Pashmina Cameron. Scaling reasoning efficiently via relaxed on-policy distillation. *arXiv preprint arXiv:2603.11137*, 2026.

Minchong Li, Feng Zhou, and Xiaohui Song. Bild: Bi-directional logits difference loss for large language model distillation. In *Proceedings of the 31st International Conference on Computational Linguistics*, pp. 1168–1182, 2025.

Yaxuan Li, Yuxin Zuo, Bingxiang He, Jinqian Zhang, Chaojun Xiao, Cheng Qian, Tianyu Yu, Huanang Gao, Wenkai Yang, Zhiyuan Liu, et al. Rethinking on-policy distillation of large language models: Phenomenology, mechanism, and recipe. *arXiv preprint arXiv:2604.13016*, 2026.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, Thomas Hubert, Peter Choy, Cyprien de Masson d'Autume, Igor Babuschkin, Xinyun Chen, Po-Sen Huang, Johannes Welbl, Sven Gowal, Alexey Cherepanov, James Molloy, Daniel J. Mankowitz, Esme Sutherland Robson, Pushmeet Kohli, Nando de Freitas, Koray Kavukcuoglu, and Oriol Vinyals. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097, 2022. doi: 10.1126/science.abq1158. URL https://www.science.org/doi/abs/10.1126/science.abq1158.

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.

Wenhan Ma, Jianyu Wei, Liang Zhao, Hailin Zhang, Bangjun Xiao, Lei Li, Qibin Yang, Bofei Gao, Yudong Wang, Rang Li, et al. Mopd: Multi-teacher on-policy distillation for capability integration in llm post-training. *arXiv preprint arXiv:2606.30406*, 2026.

MAA. American mathematics competition - amc. In *American Mathematics Competition - AMC*. URL https://maa.org/student-programs/amc/.

MAA. American invitational mathematics examination - aime. In *American Invitational Mathematics Examination - AIME 2024*, February 2024. URL https://maa.org/math-competitions/american-invitational-mathematics-examination-aime.

MAA. American invitational mathematics examination - aime. In *American Invitational Mathematics Examination - AIME 2025*, February 2025. URL https://maa.org/math-competitions/american-invitational-mathematics-examination-aime.

Eric Mitchell, Rafael Rafailov, Archit Sharma, Chelsea Finn, and Christopher D Manning. An emulator for fine-tuning large language models using small language models. *arXiv preprint arXiv:2310.12962*, 2023.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937. PmLR, 2016.

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

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candes, and Tatsunori Hashimoto. s1: Simple test-time scaling. In *Workshop on Reasoning and Planning for Large Language Models*, 2024.

NVIDIA Corporation. Opensciencereasoning-2. https://huggingface.co/datasets/nvidia/OpenScienceReasoning-2, 2025. Hugging Face dataset.

Hao Peng, Xin Lv, Yushi Bai, Zijun Yao, Jiajie Zhang, Lei Hou, and Juanzi Li. Pre-training distillation for large language models: A design space exploration. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3603–3618, 2025.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

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

Hejian Sang, Yuanda Xu, Zhengze Zhou, Ran He, Zhipeng Wang, and Jiachen Sun. Crisp: Compressed reasoning via iterative self-policy distillation. *arXiv preprint arXiv:2603.05433*, 2026.

Mingyang Song and Mao Zheng. A survey of on-policy distillation for large language models. *arXiv preprint arXiv:2604.00626*, 2026.

Zafir Stojanovski, Oliver Stanley, Joe Sharratt, Richard Jones, Abdulhakeem Adefioye, Jean Kaddour, and Andreas Köpf. Reasoning gym: Reasoning environments for reinforcement learning with verifiable rewards, 2025. URL https://arxiv.org/abs/2505.24760.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

M-A-P Team, Xinrun Du, Yifan Yao, Kaijing Ma, Bingli Wang, Tianyu Zheng, Kang Zhu, Minghao Liu, Yiming Liang, Xiaolong Jin, Zhenlin Wei, Chujie Zheng, Kaixin Deng, Shian Jia, Sichao Jiang, Yiyan Liao, Rui Li, Qinrui Li, Sirun Li, Yizhi Li, Yunwen Li, Dehua Ma, Yuansheng Ni, Haoran Que, Qiyao Wang, Zhoufutu Wen, Siwei Wu, Tianshun Xing, Ming Xu, Zhenzhu Yang, Zekun Moore Wang, Junting Zhou, Yuelin Bai, Xingyuan Bu, Chenglin Cai, Liang Chen, Yifan Chen, Chengtuo Cheng, Tianhao Cheng, Keyi Ding, Siming Huang, Yun Huang, Yaoru Li, Yizhe Li, Zhaoqun Li, Tianhao Liang, Chengdong Lin, Hongquan Lin, Yinghao Ma, Tianyang Pang, Zhongyuan Peng, Zifan Peng, Qige Qi, Shi Qiu, Xingwei Qu, Shanghaoran Quan, Yizhou Tan, Zili Wang, Chenqing Wang, Hao Wang, Yiya Wang, Yubo Wang, Jiajun Xu, Kexin Yang, Ruibin Yuan, Yuanhao Yue, Tianyang Zhan, Chun Zhang, Jinyang Zhang, Xiyue Zhang, Xingjian Zhang, Yue Zhang, Yongchi Zhao, Xiangyu Zheng, Chenghua Zhong, Yang Gao, Zhoujun Li, Dayiheng Liu, Qian Liu, Tianyu Liu, Shiwen Ni, Junran Peng, Yujia Qin, Wenbo Su, Guoyin Wang, Shi Wang, Jian Yang, Min Yang, Meng Cao, Xiang Yue, Zhaoxiang Zhang, Wangchunshu Zhou, Jiaheng Liu, Qunshu Lin, Wenhao Huang, and Ge Zhang. Supergpqa: Scaling llm evaluation across 285 graduate disciplines, 2025. URL https://arxiv.org/abs/2502.14739.

Xiaoxuan Wang, Ziniu Hu, Pan Lu, Yanqiao Zhu, Jieyu Zhang, Satyen Subramaniam, Arjun R. Loomba, Shichang Zhang, Yizhou Sun, and Wei Wang. SciBench: Evaluating College-Level Scientific Problem-Solving Abilities of Large Language Models. In *Proceedings of the Forty-First International Conference on Machine Learning*, 2024.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

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.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Wenkai Yang, Weijie Liu, Ruobing Xie, Kai Yang, Saiyong Yang, and Yankai Lin. Learning beyond teacher: Generalized on-policy distillation with reward extrapolation. *arXiv preprint arXiv:2602.12125*, 2026.

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.

APPENDIX

A RELATED WORKS

Post-training commonly relies on SFT and RL to align LLMs with reasoning tasks. On-policy distillation (OPD) has recently emerged as an alternative post-training method built on knowledge distillation. The distillation transfers the outputs of a high-performing teacher to a student (Hinton et al., 2015). In fact, the SFT above is itself a form of KD. Its first form learns from teacher-generated sequences, named Sequence-KD (Kim & Rush, 2016), and later generalized to SFT on large model-generated sequences (Muennighoff et al., 2024; Guha et al., 2025). However, training on the teacher's sequences has a limitation. The student learns from the teacher's distribution but generates from its own distribution at inference. This train-inference mismatch is known as exposure bias (Bengio et al., 2015). On-policy distillation addresses this issue. Instead of teacher sequences, it uses the student's

own rollout samples and lets the teacher score the sampled tokens (Agarwal et al., 2024; Gu et al., 2026; Ko et al., 2024). Thus, the learning signal is applied to the states the student actually visits at inference. Recent studies show that OPD is competitive with RL in reasoning post-training, with dense token-level signals and reduced cost (Lu & Lab, 2025), and its efficiency has been reported on small models in recent releases (Yang et al., 2025; Xiao et al., 2026).

Following OPD, various studies extend or analyze the on-policy distillation framework. Some studies modify the learning signal, such as reweighting it by uncertainty (Jin et al., 2026) or allowing a controlled degree of off-policy data for efficiency (Ko et al., 2026). Other studies remove the external teacher and use a single LLM as both the teacher and the student, in which the teacher gets a verified solution as privileged information while the student sees only the problem (Zhao et al., 2026; Sang et al., 2026). There are also studies that analyze OPD rather than extend it (Kim et al., 2026; Li et al., 2026; Song & Zheng, 2026). These methods share a common point. They all set the teacher's output as the target for the student. The most related work to ours uses the teacher's base model for the distillation signal. ExOPD (Yang et al., 2026) uses the teacher-base model to build an extrapolation signal. It amplifies the difference between the teacher and its base with a factor $\lambda > 1$, so the student is trained beyond the teacher. Our OPD^2 also uses the teacher base model, but in a different way. We do not extrapolate the teacher's output. Instead, we use the difference between the teacher and its base, the *delta signal*, as the main reward. The *delta signal* represents the knowledge that the teacher learned from reasoning tuning. With centering and the joint condition, OPD^2 focuses the signal on reasoning-related tokens while keeping the student's original ability.

B Additional experiments

B.1 Statistical Analysis.

We further analyze token-level signal changes using statistical values when the distillation signal is changed from OPD to Δ. For this analysis, Qwen3-8B is used for the student, Qwen3-30B-A3B-Thinking-2507 for the teacher, and Qwen3-30B-A3B-Base for the base. We select a large model setting to show that the clear signal statistics reduce noisy signals from failure reasoning traces. We use the questions from OpenMathReasoning (Moshkov et al., 2025) for the math domain, OpenCodeReasoning (Ahmad et al., 2025) for the code domain, and OpenScienceReasoning-2 (NVIDIA Corporation, 2025) for the science domain. In each domain, we randomly sample 10k questions and generate reasoning responses with the student model. The generated reasonings are evaluated on the teacher and base models and converted into distillation signals: OPD and Δ. For each token in the vocabulary, we measure how often its signal changes by at least 1 when replacing OPD with Δ. We separately count increases and decreases as enhanced and suppressed cases, respectively. The thresholds of $+1$ and -1 correspond approximately to the top and bottom 4% of OPD signal values and 5% of Δ signal values, respectively. For example, an enhanced ratio of 50% indicates that half of the occurrences of a token receive a positive signal change of at least 1. Such a token is thus strongly biased toward enhancement due to changes in the distillation signal.

Table 5 shows the words with high distillation strength changes when the on-policy distillation reward is replaced with Δ. We conduct analysis on three domains, Math, Code, and Science, where each domain contains 10k questions, and the number of generated tokens is denoted beside the domain name. Across domains, Δ tends to enhance explicit logical connection words: *hence*, *thus*, *however*, and *regardless*. In suppressed words, a vague expression of uncertainty, *perhaps*, is commonly suppressed with a high ratio. Also, Δ suppresses words frequently used in generic problem-solving narration, including *tackle*, *look*, *consider*, *analyze*, and *computing*. In Math, *note*, *why*, and *depending* are enhanced, suggesting stronger emphasis on observation, verification, and conditional reasoning, while conversational expressions such as *i'm* and *we're* are suppressed. In Code and Science, reflective expressions such as *imagine*, *oh*, *hmm*, and *another* are also more frequent, suggesting increased consideration of alternative reasoning paths.

B.2 Qwen3-8B

We additionally evaluate Qwen3-8B in the non-thinking mode, with the full results reported in Table 6. Consistent with the results for Qwen3-1.7B and Qwen3-4B in the main text, OPD^2 achieves the best average performance across all three domains. In particular, OPD^2 improves the original model

Table 5: **Distillation signal difference statistics from OPD to Δ.** The table shows vocabulary tokens that are highly biased toward being enhanced or suppressed when the distillation signal is replaced with the *delta signal*. The ratio represents the token occurrence rate with top-5% strength, and count denotes the total word count across all reasoning traces.

	Math (72.4M)			**Code** (53.3M)			**Science** (36.7M)		
Shift	Word	Ratio	Count	Word	Ratio	Count	Word	Ratio	Count
Enhanced	`hence`	57%	5.0k	`imagine`	74%	1.9k	`hence`	60%	4.9k
	`however`	55%	48.0k	`however`	61%	28.6k	`hmm`	49%	3.5k
	`note`	53%	8.7k	`thus`	60%	6.1k	`another`	45%	10.8k
	`why`	52%	2.5k	`oh`	60%	2.0k	`i've`	44%	0.8k
	`depending`	46%	2.6k	`1?),`	59%	2.1k	`lists`	43%	0.9k
	`regardless`	45%	2.9k	`regardless`	53%	1.5k	`however`	39%	46.0k
Suppressed	`perhaps`	49%	24.7k	`tackle`	73%	0.8k	`entirely`	57%	1.7k
	`i'm`	47%	1.1k	`computing`	34%	0.6k	`perhaps`	38%	20.6k
	`we're`	41%	0.7k	`suppose`	34%	1.3k	`look`	31%	1.2k
	`incorrect`	38%	0.6k	`incorrect`	32%	1.0k	`wants`	30%	0.8k
	`see`	36%	7.7k	`consider`	31%	5.6k	`analyze`	29%	0.6k
	`differently`	36%	0.7k	`denote`	31%	1.7k	`designed`	29%	1.4k

Table 6: **Qwen3-8B non-thinking results.** The table shows non-thinking mode results for Qwen3-8B. The best performance is shown in bold, and the second best is underlined.

	Math							
Model	AIME24	AIME25	AMC23	HMMT25	MATH500	Olympiad	RGMath	**Avg**
Qwen3-8B	28.3	17.9	66.2	10.7	80.5	34.2	90.1	46.9
+ OPD	62.9	46.2	90.2	<u>33.3</u>	90.0	43.7	<u>94.6</u>	65.9
+ ExOPD	<u>69.6</u>	<u>51.7</u>	<u>92.8</u>	30.0	<u>90.3</u>	<u>45.1</u>	**95.1**	<u>67.8</u>
+ OPD^2	**76.2**	**59.2**	**94.8**	**38.7**	**90.9**	**46.2**	**95.1**	**71.6**

	Code					**Science**			
Model	Code Contests	Code Forces	LCBv5	RG Algo	**Avg**	GPQA	Super GPQA	Sci Bench	**Avg**
Qwen3-8B	15.2	20.5	27.7	36.7	25.0	49.1	34.1	47.3	43.5
+ OPD	18.0	29.7	<u>44.9</u>	48.7	35.3	61.0	**38.9**	49.1	49.7
+ ExOPD	<u>30.1</u>	**33.9**	**47.4**	<u>57.8</u>	<u>42.3</u>	<u>61.8</u>	38.5	<u>50.4</u>	<u>50.2</u>
+ OPD^2	**34.1**	<u>31.1</u>	43.6	**69.4**	**44.6**	**63.8**	<u>38.6</u>	**52.3**	**51.6**

from 46.9 to 71.6 on Math, from 25.0 to 44.6 on Code, and from 43.5 to 51.6 on Science. It also consistently outperforms both OPD and ExOPD, demonstrating that the effectiveness of OPD^2 is maintained across different model scales.

B.3 Thinking mode

Tables 7 and 8 provide the full benchmark results for Qwen3 in the thinking mode. Unlike the non-thinking setting, the original models already exhibit strong reasoning capabilities, making further improvement substantially more challenging. Standard OPD often degrades the original performance, while ExOPD provides only limited or inconsistent gains. In contrast, OPD^2 achieves the highest average performance across Math, Code, and Science for Qwen3-1.7B, Qwen3-4B, and Qwen3-8B. These results show that OPD^2 can improve already strong reasoning models while better preserving their existing capabilities.

B.4 Computational costs

OPD^2 introduces additional computation compared with standard OPD because it requires a teacher-base model forward pass to construct the *delta signal*. As shown in Table 9, this increases the wall-clock training time by approximately 24-28% for the Qwen3 models and by 8% for Gemma4-E4B. For OPD^2, this additional computation represents a modest overhead compared with standard

Table 7: **Qwen3 thinking results for Math.** The table shows thinking mode results for various size of Qwen3. The best performance is shown in bold, and the second best is underlined.

Model	AIME24	AIME25	AMC23	HMMT25	MATH500	Olympiad	RGMath	Avg
Qwen3-1.7B	50.4	32.9	85.2	22.3	86.6	39.9	96.8	59.2
+ OPD	41.5	34.2	80.2	20.7	85.6	39.9	97.3	57.1
+ ExOPD	44.6	35.4	84.5	20.3	86.6	40.6	96.9	58.4
+ OPD^2	**51.9**	**42.7**	**89.8**	**26.3**	**88.1**	**42.1**	**97.7**	**62.7**
Qwen3-4B	**73.3**	64.8	96.2	44.3	90.9	45.4	98.1	73.3
+ OPD	65.4	56.0	98.0	41.7	**91.9**	45.8	97.6	70.9
+ ExOPD	68.1	60.2	98.0	43.7	91.8	46.0	**98.5**	72.3
+ OPD^2	**73.3**	**66.9**	**98.2**	**48.7**	91.5	**47.3**	97.7	**74.8**
Qwen3-8B	76.0	64.0	95.8	44.3	91.0	46.3	98.4	73.7
+ OPD	69.2	59.4	97.2	43.3	91.5	46.9	98.2	72.2
+ ExOPD	71.7	62.1	97.5	46.3	**91.9**	47.1	**98.7**	73.6
+ OPD^2	**76.5**	**66.9**	**98.8**	**52.3**	**91.9**	**47.5**	97.7	**75.9**

Table 8: **Qwen3 thinking results for Code and Science.** The table shows thinking mode results for Qwen3. The best performance is shown in bold, and the second best is underlined.

	Code					Science			
Model	Code Contests	Code Forces	LCBv5	RG Algo	Avg	GPQA	Super GPQA	Sci Bench	Avg
Qwen3-1.7B	19.8	21.3	32.6	44.3	29.5	48.0	28.1	**44.1**	40.1
+ OPD	25.9	24.0	35.1	45.2	32.6	54.2	29.4	41.6	41.7
+ ExOPD	36.0	28.7	39.7	44.6	37.2	54.3	29.6	42.6	42.2
+ OPD^2	**38.0**	**30.5**	**42.6**	**51.8**	**40.7**	**56.8**	**30.2**	43.4	**43.5**
Qwen3-4B	34.8	39.7	59.4	64.1	49.5	**58.2**	35.6	**60.1**	51.3
+ OPD	38.6	38.6	55.1	64.1	49.1	51.4	35.6	58.1	48.4
+ ExOPD	42.6	41.6	56.3	67.8	52.1	52.9	36.0	58.1	49.0
+ OPD^2	**45.3**	**45.5**	**62.6**	**73.4**	**56.7**	57.5	**38.1**	58.8	**51.5**
Qwen3-8B	39.2	42.3	51.0	72.4	51.2	64.1	37.2	57.8	53.0
+ OPD	41.0	43.0	59.1	66.5	52.4	59.3	37.2	56.3	50.9
+ ExOPD	46.5	44.0	63.3	69.8	55.9	61.1	38.3	56.5	52.0
+ OPD^2	**48.5**	**48.7**	**67.5**	**73.7**	**59.6**	**64.4**	**40.6**	**58.7**	**54.6**

OPD. However, the computation cost of OPD^2 remains comparable to that of ExOPD, with only a small difference in training time across all model sizes. Our current implementation is not optimized for the reward computation in Eq. 4, suggesting that the reported overhead can be further reduced. Moreover, on-policy distillation typically converges within a relatively small number of training steps, and prolonged training can even lead to performance degradation, as observed in Fig. 4. Thus, although OPD^2 requires additional computational cost compared to standard OPD, its additional overhead remains limited in the typical short-training regime of on-policy distillation.

C Experimental details

Tables 10 and 11 summarize the model-specific configurations and common optimization settings used in our experiments. We use NVIDIA H100 GPUs for all experiments. Qwen3-1.7B is trained on a single 8-GPU node, while Qwen3-4B, Qwen3-8B, and Gemma4-E4B are trained using four nodes. For the multi-node settings, 24 GPUs are used for student training and 8 GPUs are allocated to the vLLM rollout server.

For Qwen3, we use a larger model from the same model family as the teacher. The Qwen3-1.7B student uses Qwen3-4B as its teacher, while the Qwen3-4B and Qwen3-8B students use Qwen3-30B-A3B. We select the corresponding Instruct or Thinking checkpoint according to the target generation mode. Gemma4-E4B-it is evaluated only in the thinking setting and uses Gemma4-31B-it

Table 9: **Training time for on-policy distillation.** The table shows wall-clock training time (hours) spent on distillation. Parentheses show the relative increase over OPD. The time is measured on NVIDIA H100 8-GPU nodes. †Qwen3-1.7B runs on a single node, while all others run on 4 nodes.

Method	Qwen3-1.7B†	Qwen3-4B	Qwen3-8B	Gemma4-E4B
OPD	4.4 (+0%)	7.3 (+0%)	7.6 (+0%)	12.7 (+0%)
ExOPD	5.3 (+19%)	9.1 (+26%)	9.4 (+24%)	13.8 (+9%)
OPD^2	5.5 (+24%)	9.3 (+28%)	9.6 (+27%)	13.8 (+8%)

as the teacher. The corresponding pretrained checkpoints without instruction tuning are used as the teacher-base models for ExOPD and OPD^2.

All methods are trained for 100 optimization steps with an effective global batch size of 256 or 288. We use AdamW with a learning rate of 5×10^{-6}, cosine lr decay, a warmup ratio of 0.1, and a minimum learning-rate ratio of 0.1. The gradient norm is clipped to 1.0, and all the rewards are scaled by 0.1 to prevent excessive gradient clipping. We set the maximum generation length to 8192 and the sampling temperature to 0.7. The KL coefficient is set to $\beta = 0$, as we do not apply an additional KL regularization term with the reference model during training.

Table 10: **Model configuration & model-specific setting.** The table shows teacher and student model configuration with model-specific parameters. We use NVIDIA H100 GPU for all experiments.

Setting	Qwen3-1.7B	Qwen3-4B / -8B	Gemma-4-E4B
Device	1 node, 8 GPUs	4 nodes, 24+8 GPUs	4 nodes, 24+8 GPUs
Student	Qwen3-1.7B	Qwen3-4B, Qwen3-8B	Gemma-4-E4B-it
Teacher (non-thinking)	Qwen3-4B-Instruct-2507	Qwen3-30B-A3B-Instruct-2507	—
Teacher (thinking)	Qwen3-4B-Thinking-2507	Qwen3-30B-A3B-Thinking-2507	Gemma-4-31B-it
Teacher-base	Qwen3-4B-Base	Qwen3-30B-A3B-Base	Gemma-4-31B
Thinking mode	non-thinking + thinking	non-thinking + thinking	thinking
vLLM rollout mode	colocate (TP=4)	server node (TP=8)	server node (TP=8)
Effective global batch	256	288	288

Table 11: **Common training parameters.** Training parameters shared across all models.

Setting	Value
Learning rate	5×10^{-6}
LR schedule	cosine decay
Minimum LR ratio	0.1
Warmup ratio	0.1
Optimizer	AdamW
Gradient clipping	1.0
Training steps	100
Maximum generation length	8192
Sampling temperature	0.7
KL coefficient β	0.0
Gradient multiplier	0.1

Table 12 summarizes the evaluation configurations for all benchmarks. We report pass@1, interpreted as single-sample accuracy, and average the scores over multiple independent repetitions to reduce the variance introduced by stochastic generation. The repetitions are not used for best-of-k selection. AIME24 and AIME25 are evaluated with 16 repetitions because of their relatively small number of problems, whereas larger benchmarks are evaluated with fewer repetitions.

Table 12: **Evaluation benchmark configs.** For each benchmark, we report pass@1 (single-sample accuracy), averaged across the listed number of repetitions.

Domain	Benchmark	Repetitions
Math	AIME24	16
	AIME25	16
	AMC23	10
	HMMT	10
	MATH500	10
	OlympiadBench	3
	RGMath	3
Code	LiveCodeBenchv5	3
	CodeForces	3
	CodeContests	3
	RGAlgorithmic	3
Science	GPQADiamond	10
	SuperGPQA	3
	SciBench	3