WEAK-TO-STRONG ON-POLICY DISTILLATION

Anonymous authors
Paper under double-blind review

ABSTRACT

On-policy distillation (OPD), which aligns a student with the teacher's token-level distribution on the student's own rollouts, has become an effective paradigm for transferring capabilities across large language models (LLMs). Prevailing approaches assume a teacher at least as capable as the student, and either distill a larger model into a smaller one, which fails at the frontier when no larger teacher exists, or train multiple domain experts from a shared base and consolidate them into one student, which requires costly training at the student's scale. To tackle these challenges, we introduce Weak-to-Strong On-Policy Distillation (W2S-OPD), a simple yet effective OPD framework that improves the strong student by distilling from multiple weak models. Specifically, W2S-OPD constructs a proxy teacher in logit space from a contrast pair of a positive and a negative model, both smaller than the student and cheap to obtain. Their logit difference isolates the capability direction, which is then added to the student's own base model. The resulting proxy teacher thus couples this direction while staying distributionally adjacent to the student. The student then distills it by minimizing the per-token reverse KL on its own rollouts. We instantiate the contrast pair as i) a post-RL expert against its pre-RL initialization, isolating the skill RL instills, ii) a larger against a smaller base model, isolating the capability from scale, and iii) a small base model with correct and wrong hints, isolating the instance-level direction toward the solution. Across four math and three code benchmarks, W2S-OPD consistently outperforms OPD and even enables the student to surpass the domain teacher and continues to improve the student when every supervision source is weaker, with gains holding from small dense models (e.g., 8B) to large MoE models (e.g., 30B). Further analysis shows that different contrasts yield distinct learning signals: the post-RL and hint contrasts emphasize reasoning frameworks, while the scale contrast emphasizes the solving procedure.

1 INTRODUCTION

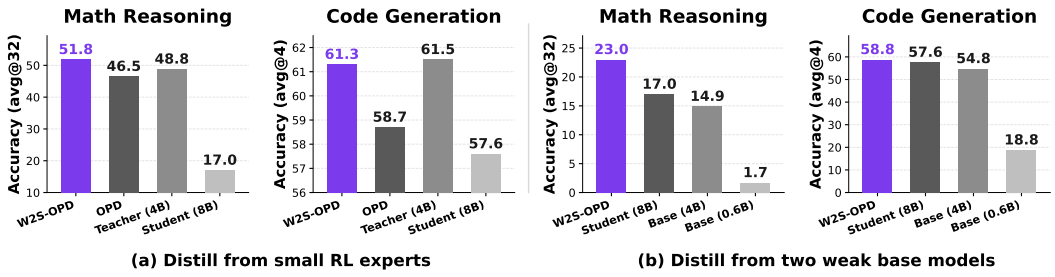


Figure 1: **W2S-OPD improves Qwen3-8B using 4B and 0.6B models as teachers.** Accuracy is averaged over 4 math reasoning and 3 code generation benchmarks. (a) With a post-RL Qwen3-4B expert, W2S-OPD beats OPD. (b) From 2 off-the-shelf base models (Qwen3-4B and 0.6B), both weaker than the student and used without training, W2S-OPD still lifts the student above itself and both sources.

Reinforcement learning with verifiable rewards (RLVR) (Guo et al., 2025; Shao et al., 2024; Yu et al., 2024) and knowledge distillation (Gu et al., 2024; Xiao et al., 2026) are two dominant paradigms for improving the reasoning ability of large language models (LLMs). RLVR scales optimization

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.

directly from verifiable outcomes, yet its sparse reward provides limited fine-grained supervision. In contrast, knowledge distillation offers dense token-level supervision from a teacher, but the learning from teacher-generated off-policy trajectories suffers from exposure bias. On-policy distillation (OPD) (Agarwal et al., 2024; Lu & Lab, 2025) combines the complementary strengths of both, which supervises the student with a teacher's token-level fine-grained supervision, yet on trajectories sampled from the student's own policy. This approach delivers dense credit assignment that alleviates exposure bias. However, the premise is a teacher at least as capable as the student.

This premise breaks down in the two ways current practice makes concrete. The strong-to-weak paradigm distills a larger model into a smaller one. For instance, Qwen3 (Yang et al., 2025) distills a large-scale model (*e.g.,* 235B) into smaller variants (*e.g.,* 8B, 14B). As models approach the frontier, however, no larger teacher exists to distill from, capping the performance ceiling. A second line instead performs same-origin consolidation: rather than assuming a single superior teacher, it trains multiple domain experts in parallel from a shared base model and distills their capabilities back into one student. MOPD (Ma et al., 2026) instantiates this by applying per-domain RL to obtain a set of domain teachers and merging them in policy space, while DeepSeek-V4 (Xu et al., 2026) scales the same specialize-then-consolidate recipe to the trillion-parameter regime. Yet training multiple domain experts at the student's scale is computationally expensive, inflating the cost of OPD. We therefore move to the weak-to-strong paradigm (Burns et al., 2023; Ning et al., 2024), which induces supervisory signals from weaker models that already exist or can be trained inexpensively. This enables a strong model to keep improving when no stronger teacher is available at a lower cost.

However, directly applying OPD to weak-to-strong learning is challenging. First, the weak teacher's distribution is distinct from the student's, and such a mismatch weakens the distillation signals (Ko et al., 2026; Li et al., 2026c). Second, since the teacher is weaker than the student, distillation toward it bounds the student at the teacher's level and risks eroding the general capability it already possesses.

To address these challenges, we propose Weak-to-Strong On-Policy Distillation (W2S-OPD), a simple yet effective OPD framework that distills the strong student from multiple weak models. Figure 2 provides an overview. Inspired by decoding-time controlled generation (Liu et al., 2021; 2024), W2S-OPD constructs a proxy teacher in logit space from a contrast pair of a positive model and a negative model, both substantially smaller than the student and cheap to obtain. Their logit difference isolates the capability direction, and W2S-OPD adds this direction to the student's own base model. The proxy teacher therefore inherits the isolated direction while remaining distributionally close to the student. The student then distills it by minimizing the per-token reverse KL on its own rollouts.

We propose three ways to construct the proxy teacher from weak models: i) **pre-RL and post-RL models**, where the positive model is a domain expert trained by RL and the negative model is its pre-RL initialization, whose difference isolates the skill that RL instills; since the RL needs only be run at the small model's scale, this transfers an expensive capability to the large student without ever running RL at the student's scale; ii) **larger and smaller base models**, two off-the-shelf base models of different sizes, whose difference isolates the capability that emerges from scale; this signal comes for free from models that already exist, requiring no new data, reward design, or training; and iii) **a single base model conditioned on correct and wrong hints**, which extracts the instance-level direction toward the solution from single model and provides token-level supervision more efficiently. We evaluate all three settings on four math reasoning and three code generation benchmarks. Our main findings are threefold:

- **Surpassing the teacher it learns from.** In the pre-RL / post-RL setting, W2S-OPD outperforms OPD by 11.4% and 12.0% relative on math reasoning under single- and multi-teacher distillation, and lifts the student above the domain expert itself. This advantage persists when scaling up to a 30B MoE model.
- **Improving from purely weaker models.** In the smaller / larger and contrastive-hints settings, W2S-OPD still improves the student even though every supervision source is weaker than it.
- **Different contrasts reinforce complementary reasoning patterns.** The post-RL and hint contrast places more emphasis on the tokens for reasoning framework (*e.g.,* planning and monitoring the solving process), whereas the scale contrast emphasizes the solving procedure.

More broadly, W2S-OPD reframes weak-to-strong learning as isolating and transferring a capability direction rather than imitating a weak supervisor, suggesting how frontier models can keep improving from abundant existing weak signals rather than waiting for a stronger teacher to be built.

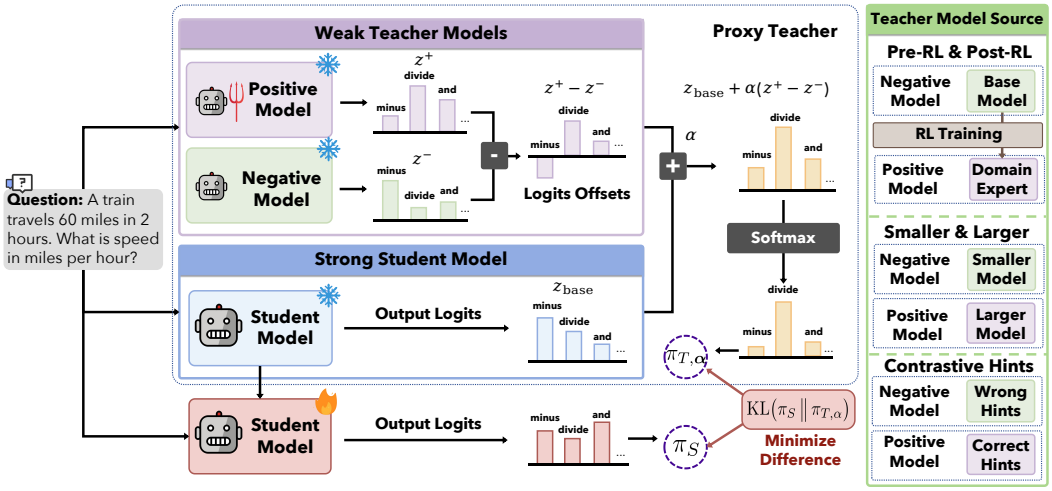


Figure 2: Overview of W2S-OPD. W2S-OPD synthesizes a proxy teacher and distills it into a student. A positive and a negative model form a contrast pair whose logit difference isolates a capability direction. The student π_S, initialized from the anchor base model, generates on-policy rollouts and minimizes the per-token reverse KL toward this proxy teacher. The contrast pair can be instantiated as a post-RL expert against its pre-RL initialization, a larger base model against a smaller one, or a single model conditioned on correct against wrong hints.

2 Preliminary: On-Policy Distillation

On-policy distillation (OPD), formalized by the Generalized Knowledge Distillation (GKD) framework (Agarwal et al., 2024), bridges reinforcement learning and imitation learning by providing dense, token-level supervision on the student's own trajectories. Unlike off-policy distillation that learns from a fixed teacher-generated dataset, OPD lets the student policy π_S generate rollouts on-policy, and a powerful frozen teacher π_T then scores every token of these self-generated trajectories, against which the student minimizes a per-token divergence:

$$\mathcal{L}_{\mathrm{OPD}}(\pi_S) \;=\; \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_S(\cdot|x)}\left[\sum_{t=1}^{|y|} D\big(\pi_S(\cdot \mid s_t)\,\big\|\,\pi_T(\cdot \mid s_t)\big)\right], \tag{1}$$

where $s_t = (x, y_{<t})$ is the student-generated prefix and D is a per-token divergence. Because states are drawn from the student's own rollouts, the teacher corrects the student precisely on the states the student actually visits, thereby avoiding the exposure bias of off-policy distillation. We instantiate D as the reverse KL divergence $\mathrm{KL}(\pi_S \,\|\, \pi_T)$. In practice, we employ Top-k OPD for computational efficiency; see more details in Appendix B.

Two prominent uses of OPD are strong-to-weak distillation and same-origin consolidation. The former compresses a larger, more capable teacher into a smaller student (Yang et al., 2025), while the latter distills several RL-trained domain experts back into a single model, with all experts initialized from the student's base so that their distributions stay close (Xiao et al., 2026; Ma et al., 2026). Both presuppose a teacher at least as capable as the student, which becomes ineffective when training the frontier model, since no larger teacher exists to distill from, and training qualified experts at the student's scale is computationally expensive. These limitations motivate us to instead leverage small, cheaply obtained models to supervise a stronger student.

3 Weak-to-Strong On-Policy Distillation

Improving a strong student with supervision from weaker models presents two central challenges: i) the distributional mismatch between the weak models and the student renders direct distillation ineffective (Ko et al., 2026; Li et al., 2026c); and ii) since the supervision sources are weaker than the student, direct distillation constrains the student to a lower-capability policy and degrades the

general ability it already possesses. To address both challenges, we adopt a directional perspective, as illustrated in Figure 2. Instead of imitating a weak model directly, we isolate the capability direction that separates a stronger positive model from a weaker negative model, a signal that is largely transferable from model scale and hence transferable across scales, and re-anchor this direction onto the student's base model. The resulting proxy teacher is simultaneously equipped with the target domain capability and distributionally adjacent to the student, which stabilizes optimization and preserves the student's general ability. We detail the proxy teacher construction and distillation objective in §3.1, the instantiations of the contrast pair in §3.2, and its extension to multi-teacher distillation in §3.3.

3.1 Proxy Teacher Synthesis and Distillation

Suppose the student π_S is initialized from a strong base model, while the available weak models form a contrast pair: a positive model m^+ and a weaker negative model m^-. For example, m^+ can be a post-RL expert and m^- its pre-RL initialization, two base models of different sizes, or a single model conditioned on correct versus wrong hints (§3.2). W2S-OPD constructs the teacher directly from the frozen base model and the contrast pair, requiring only their output logits.

The key observation behind W2S-OPD is that, although both m^+ and m^- are weaker than the strong base model, their difference still encodes a transferable capability direction. What the two weak models agree on reflects their shared, limited ability, so subtracting their logits cancels this common component and retains precisely the direction along which m^+ improves over m^-. Adding this capability direction onto the strong base model therefore yields a proxy teacher that combines it with the student's general strength while staying distributionally close to the student. We instantiate this construction following decoding-time experts (Liu et al., 2021; 2024), where m^+ acts as the positive model whose logits are additively combined, m^- acts as the negative model whose logits are negatively combined, and the student's base model serves as the anchor. Formally, at each position t, we condition the three models on the prefix s_t to obtain the logit scores z_{base}, z^+, and z^-, and synthesize the proxy teacher as:

$$\pi_{T,\alpha}(\cdot \mid s_t) \;=\; \text{softmax}\Big(z_{\text{base}}(s_t) \;+\; \alpha\big(z^+(s_t) - z^-(s_t)\big)\Big), \tag{2}$$

where $\alpha \geq 0$ is an amplification coefficient controlling the strength of the injected capability direction, and thus trades off signal strength against distributional proximity. A small α injects little of the direction and keeps the proxy teacher close to the base model, yielding supervision too weak to drive improvement, whereas a large α injects a stronger signal but risks distorting the distribution and pushing the teacher away from the student. Given the constructed proxy teacher, W2S-OPD instantiates the OPD objective in Eq. 1 with $\pi_T = \pi_{T,\alpha}$:

$$\mathcal{L}_{\text{W2S-OPD}}(\pi_S) \;=\; \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_S(\cdot|x)}\left[\sum_{t=1}^{|y|} \text{KL}\big(\pi_S(\cdot \mid s_t) \,\big\|\, \pi_{T,\alpha}(\cdot \mid s_t)\big)\right]. \tag{3}$$

Together, W2S-OPD addresses both challenges raised at the beginning of this section. Anchoring the proxy teacher at the student's own base model keeps the supervision target within a distribution the student already realizes (with α bounding how far it moves), and since the weak models enter only through their logit difference rather than their absolute level, distillation transfers their capability direction without pulling the student down toward them, which alleviates the capability ceiling. Appendix C gives an alternative view of this objective as reward maximization under a KL trust region.

3.2 Instantiations of the Positive and Negative Models

W2S-OPD requires a contrast pair (m^+, m^-) to construct the proxy teacher. We propose three instantiations and summarize in Table 1:

i) *Pre-RL and post-RL*: the positive model m^+ is a domain expert obtained by applying RL to a small base model, and the negative model m^- is its pre-RL initialization, whose difference isolates the domain skill acquired through RL. This avoids the cost of training domain experts at the student's own scale.

Table 1: The three instantiations of the contrast pair in W2S-OPD. All models are from Qwen3 series.

Setting	**Positive model** m^+	**Negative model** m^-	**Anchor** z_{base}
Pre-RL / Post-RL	4B-RL (post-RL expert)	4B (pre-RL init)	8B (base)
Smaller / Larger	4B (base)	0.6B (base)	8B (base)
Correct / Wrong Hints	4B w/ correct hint	4B w/ wrong hint	8B (base)

ii) *Smaller and larger*: the positive and negative models are two off-the-shelf base models of different sizes (*e.g.,* Qwen3-4B and Qwen3-0.6B). Their difference isolates the capability that emerges from scale and requires no additional training. This signal is already available in released models and can be used for free to further improve the capability of frontier models.

iii) *Correct and wrong hints*: the positive and negative models are a single base model conditioned on a correct and a wrong solution hint of the identical format, respectively. Since the hint-conditioning is shared, their difference cancels the style shift it induces (Pan et al., 2026) and isolates the instance-level direction toward the correct solution. Needing only one small model and a reference solution provides token-level supervision efficiently.

These instantiations indicate that W2S-OPD can support various origins of the contrast, in which the pair exhibiting a separable capability gap can serve as the supervision source, since the construction of the proxy teacher requires only the output logits of the pair.

3.3 Unification of Single- and Multi-Proxy Teacher Distillation

W2S-OPD can naturally extend to multiple proxy teacher distillation scenarios. Given K positive models $\{m_k^+\}_{k=1}^K$, each paired with a weaker negative model m_k^-, W2S-OPD composes their capability directions on the shared base model:

$$\pi_{T,\boldsymbol{\alpha}}(\cdot \mid s_t) \;=\; \text{softmax}\Big(z_{\text{base}}(s_t) \;+\; \sum_{k=1}^{K} \alpha_k \big(z_k^+(s_t) - z_k^-(s_t) \big) \Big), \tag{4}$$

where α_k controls the strength of capability direction k. Since each subtraction isolates its own capability direction, the summation injects multiple domain skills into a single proxy teacher, and one distillation run with Eq. 3 merges them into a unified student. For example, α_k can be set to $\{0, 1\}$. In this way, each query is routed to its most suitable positive and negative model pair for distillation.

4 Experiments

We evaluate W2S-OPD under the three contrast-pair settings. §4.2 reports the main results for each, §4.3 examines the factors behind its effectiveness, and §4.4 characterizes the learning signals the contrast provides.

4.1 Experimental Setups

Benchmarks. For math reasoning, we evaluate on four competition-level benchmarks: AIME24 (AI-MO, 2024), AIME25 (Zhang & Math-AI, 2025), HMMT25 (Feb.), and HMMT25 (Nov.) (Balunovic et al., 2025). For code generation, we adopt HumanEval+, MBPP+ (Liu et al., 2023), and LiveCodeBench-V6 (Jain et al., 2025). To assess out-of-domain generalization, we further evaluate on GPQA-Diamond (Rein et al., 2023) for scientific reasoning and IFBench (Pyatkin et al., 2026) for instruction following.

Baselines and Evaluation Metrics. We compare W2S-OPD against OPD, which distills the student directly from the positive model, and SFT, which fine-tunes the student on the same number of trajectories rolled out from the positive model, under an identical training configuration. As reference points, we report the positive and negative models that form the contrast pair, as well as the student itself. In all evaluations, we set the temperature to 1.0 and top-p to 1.0. On each math reasoning benchmark, we sample 32 solutions per problem, whereas on each code generation benchmark, we sample 4 solutions per problem, and report the average accuracy over all samples on each benchmark for a robust evaluation. We adopt Math-Verify to validate answer correctness for math reasoning.

Table 2: Results for Pre-RL / Post-RL contrast setting. W2S-OPD beats OPD and even surpasses the 4B-expert on math. *Improv.* reports the absolute gain over OPD.

Method	Math Reasoning					Code Generation			
	AIME24	AIME25	HMMT25 (Feb.)	HMMT25 (Nov.)	Avg.	HumanEval+	MBPP+	LCB	Avg.
Neg. Model (4B)	21.6	19.9	10.0	8.2	14.9	79.9	63.6	20.9	54.8
Pos. Model (4B-RL)	62.0	57.9	34.8	40.4	48.8	86.3	71.0	27.1	61.5
Student Base (8B)	26.1	21.1	11.6	9.0	17.0	81.2	70.0	21.7	57.6
Single-Teacher Distillation									
SFT	60.2	54.4	30.7	37.6	45.7	82.5	71.8	24.8	59.7
OPD	62.1	54.8	30.6	38.3	46.5	83.1	71.2	22.0	58.7
W2S-OPD	**68.9**	**60.1**	**35.1**	**43.1**	**51.8**	**84.8**	**72.4**	**26.8**	**61.3**
Improv.	**+6.8**	**+5.3**	**+4.5**	**+4.8**	**+5.3**	**+1.7**	**+1.2**	**+4.8**	**+2.6**
Multi-Teacher Distillation									
SFT	63.5	54.6	29.6	39.4	46.8	84.0	71.4	25.7	60.4
OPD	64.3	54.9	29.6	37.4	46.5	83.4	69.5	25.6	59.5
W2S-OPD	**67.4**	**61.0**	**34.7**	**45.2**	**52.1**	**85.2**	**71.0**	**27.1**	**61.1**
Improv.	**+3.1**	**+6.1**	**+5.1**	**+7.8**	**+5.6**	**+1.8**	**+1.5**	**+1.5**	**+1.6**

Table 3: Results for Smaller and Larger contrast setting. W2S-OPD improves the 8B student above its own base even though both source models are weaker than it. *Improv.* reports the absolute gain of W2S-OPD over the student base model.

Method	Math Reasoning					Code Generation			
	AIME24	AIME25	HMMT25 (Feb.)	HMMT25 (Nov.)	Avg.	HumanEval+	MBPP+	LCB	Avg.
Neg. Model (0.6B)	1.6	2.3	0.1	2.7	1.7	25.6	26.3	4.6	18.8
Pos. Model (4B)	21.6	19.9	10.0	8.2	14.9	79.9	63.6	20.9	54.8
Student Base (8B)	26.1	21.1	11.6	9.0	17.0	81.2	70.0	21.7	57.6
W2S-OPD	**33.1**	**25.9**	**13.0**	**20.1**	**23.0**	**81.7**	**71.1**	**23.7**	**58.8**
Improv.	**+7.0**	**+4.8**	**+1.4**	**+11.1**	**+6.0**	**+0.5**	**+1.1**	**+2.0**	**+1.2**

Training details. We use Qwen3-8B (Yang et al., 2025) in non-thinking mode as the student π_S; a frozen copy of the same checkpoint serves as the anchor. Both W2S-OPD and OPD are trained for 100 steps under an identical configuration. See Appendix D for more implementation details.

Contrast pairs. Table 1 summarizes the contrast pairs of each setting. In the pre-RL / post-RL setting, the positive model m^+ is a Qwen3-4B domain expert obtained by applying GRPO (Shao et al., 2024) to the Qwen3-4B base, for 500 steps on the level-6 subset of DeepMath-103K (He et al., 2025) for math reasoning and for 300 steps on Eurus-RL-Code (Cui et al., 2025) for code generation, which are provided by Yang et al. (2026b), and the negative model m^- is its pre-RL initialization, *i.e.,* the original Qwen3-4B model. In the smaller/larger setting, the positive and negative models are the off-the-shelf Qwen3-4B and Qwen3-0.6B base models, requiring no additional training. In the contrastive-hint setting, the positive and negative models share the Qwen3-4B base model, conditioned on a correct and a wrong solution hint of the identical format, respectively.

4.2 Main Results

i) Distillation from Pre-RL / Post-RL. Table 2 compares W2S-OPD against OPD under single-teacher distillation, which learns from a single domain expert, and multi-teacher distillation, which merges multiple domain experts (here a math and a code expert) into one student. We have the following observations: **(a) A small post-RL expert substantially improves the large student.** W2S-OPD consistently outperforms SFT and OPD across all benchmarks in the single-teacher setting, with 11.4% and 4.4% average relative improvements on math and code, respectively. Notably, on math reasoning W2S-OPD surpasses the domain teacher itself, whereas OPD remains below it. On code generation, W2S-OPD likewise narrows the gap to the teacher. Results in §E.4 further demonstrate the advantage when scaling up to 30B models. **(b) W2S-OPD outperforms OPD when distilling from multiple domain teachers.** We conduct experiments in the multi-teacher setting, where we aim to merge the capabilities from different domain teachers into a single student through OPD, where each teacher is obtained by applying domain-specific RL to the same base model. Specifically, we mix the math and coding training data and route each sample to the corresponding domain teacher through their domain labels. The results in Table 2 demonstrate that W2S-OPD consistently leads to better performance than OPD on all benchmarks. **(c) W2S-OPD learns faster**

Table 4: Results for contrastive hints setting. W2S-OPD improves the 8B student above its own base with only a single weaker and smaller model. *Improv.* reports the absolute gain of W2S-OPD over the student base model.

Method	Math Reasoning					Code Generation			
	AIME24	AIME25	HMMT25 (Feb.)	HMMT25 (Nov.)	Avg.	HumanEval+	MBPP+	LCB	Avg.
Pos./Neg. Model (4B)	21.6	19.9	10.0	8.2	14.9	79.9	63.6	20.9	54.8
Student Base (8B)	26.1	21.1	11.6	9.0	17.0	81.2	70.0	21.7	57.6
W2S-OPD	**27.7**	**23.3**	**12.0**	**10.7**	**18.4**	**82.5**	**70.8**	**22.7**	**58.7**
Improv.	**+1.6**	**+2.2**	**+0.4**	**+1.7**	**+1.4**	**+1.3**	**+0.8**	**+1.0**	**+1.1**

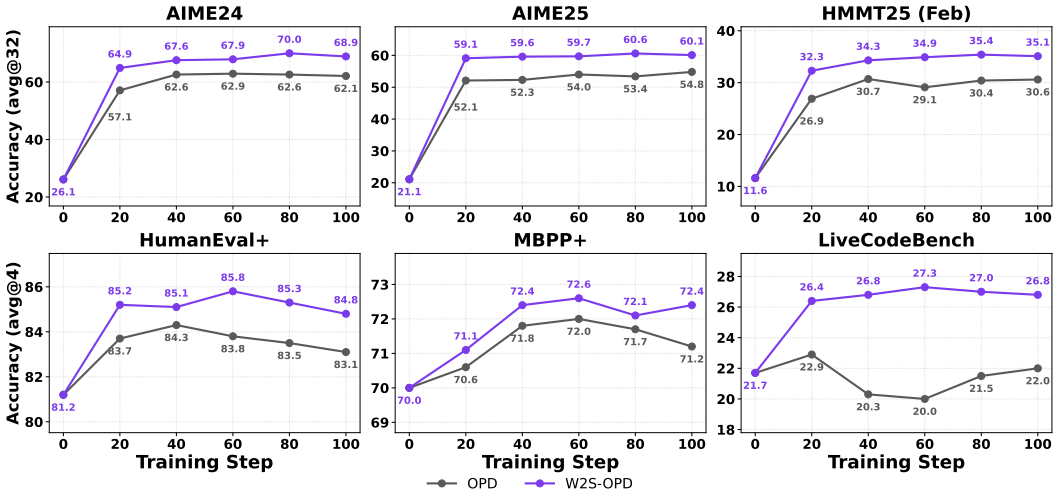


Figure 3: Performance on the math and code benchmarks over training steps. W2S-OPD improves faster and outperforms OPD.

and more stably. Figure 3 tracks benchmark performance over training and shows that W2S-OPD outperforms OPD throughout the training.

ii) Distillation from Smaller / Larger Base Models. In this setting, we find that **two weak base models can improve the stronger student.** As shown in Table 3, when the proxy teacher is constructed from Qwen3-4B and Qwen3-0.6B, both weaker than the student, W2S-OPD still improves the student by an absolute 6.0% on math reasoning and 1.2% on code generation on average. This confirms that the inherent gap between two off-the-shelf weak models also encodes a transferable improving direction to distill from.

iii) Distillation from a Single Model with Correct / Wrong Hints. As shown in Table 4, a contrastive hint direction from a single weak model improves the student. W2S-OPD improves the student by an absolute 1.4% on math reasoning and 1.1% on code generation on average, despite the hint model being a 4B model weaker than the student. This construction distills the privileged information and shows that a meaningful learning signal can also be extracted from a difference in context without requiring two different models.

4.3 Training Analysis

Using the Pre-RL / Post-RL setting, we conduct two further analyses: the impact of the amplification coefficient α on distillation, and out-of-domain generalization compared with OPD.

4.3.1 Effect of the Amplification Coefficient α

To investigate the impact of the amplification coefficient, we vary α with all other configurations fixed and report the results in Figure 4. When α is small, $\pi_{T,\alpha}$ stays close to the student's own base model, where the capability direction is barely injected, the distillation signal is weak, and the distilled student falls behind OPD. As α increases, more domain skill is injected into the proxy teacher, and

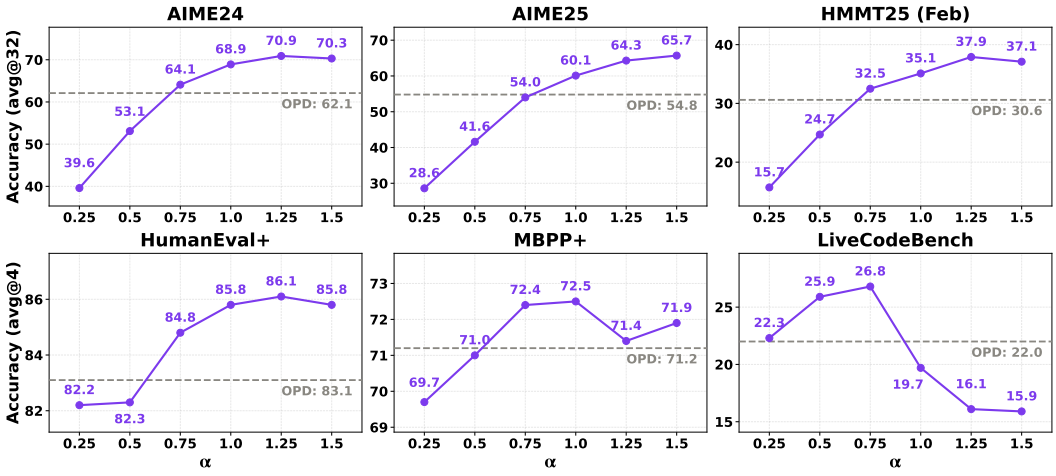


Figure 4: Performance on the math and code benchmarks with different α. OPD is included for reference, denoted by the gray line. W2S-OPD outperforms OPD over a wide range of α.

the student's performance improves accordingly. As α continues to increase, performance starts to decline on most benchmarks, likely because an overly large α drives the proxy teacher further away from the student's distribution, making the supervision increasingly hard for the student to learn. Consequently, a moderate α strikes the best balance between the strength of the injected signal and the adjacency of the teacher.

4.3.2 Generalization to Out-of-Domain Tasks

A crucial concern for weak-to-strong distillation is whether learning from a small domain expert erodes the strong student's general capability. To investigate this, we evaluate the student trained in the Pre-RL / Post-RL setting on two out-of-domain benchmarks. As shown in Table 5, the distilled reasoning skill transfers well beyond the training domain. W2S-OPD lifts the student from 38.9 to 56.5 on GPQA-Diamond, outperforming OPD by an absolute 2.1 %. The two models diverge on IFBench, where OPD degrades the student below its initialization, consistent with the student absorbing the small expert's limitations, yet W2S-OPD instead improves compared to the base model and outperforms OPD by 1.1%. These results indicate that W2S-OPD transfers domain skill without sacrificing and even enhancing the student's general ability.

Table 5: Results for OOD generalization on GPQA-Diamond and IFBench. Both distillation methods are trained only on the math task, W2S-OPD transfers out of domain and improves general ability, whereas OPD can degrade it below the base. *Improv.* indicates the absolute gain over OPD.

Method	GPQA-Diamond	IFBench
Positive Model	51.8	26.0
Student Base	38.9	26.3
OPD	54.4	25.9
W2S-OPD	**56.5**	27.0
Improv.	**+2.1**	**+1.1**

4.4 What Kinds of Tokens Are Most Strengthened?

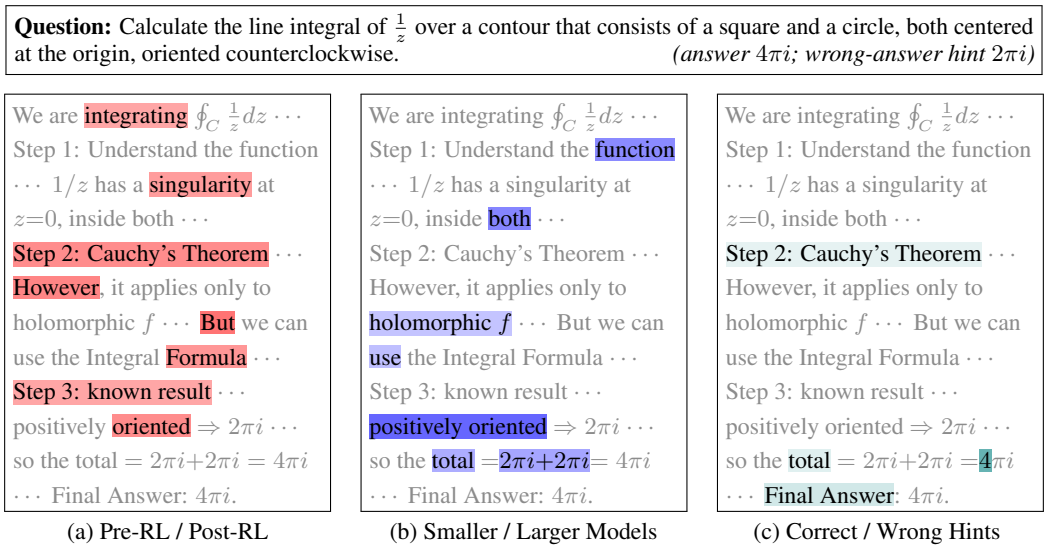
Since all settings improve the student, we raise the question of what each capability direction actually provides. We quantitatively analyze this question by recording each token's offset between the positive and negative models on a shared solution generated by the student model. At every position t along this trace, the capability direction in logit space is $\Delta z_t = z^+(s_t) - z^-(s_t)$. To measure how strongly it reinforces the realized token x_t, we take the log-probability each model assigns to x_t and define:

$$\Delta_t \;=\; \log \pi^+(x_t \mid s_t) \;-\; \log \pi^-(x_t \mid s_t), \qquad \pi^{\pm} = \mathrm{softmax}(z^{\pm}). \tag{5}$$

Tokens with large Δ_t are those most strongly reinforced by the capability direction. To identify which reasoning steps each contrast reinforces, we collect the top-1% highest-Δ tokens across 200

Table 6: Distribution of the top-1% highest-Δ tokens over the eight Schoenfeld episodes. The first row gives the distribution of each episode of all tokens.

	Read	Analyze	Plan	Implement	Explore	Verify	Monitor	Answer
All tokens	10.3	33.2	8.6	40.7	0.5	2.4	0.5	3.8
Pre-RL / Post-RL	9.6	40.5	15.7	24.0	1.2	4.3	1.8	2.9
Smaller / Larger	6.5	44.4	12.6	27.7	0.8	4.2	0.9	2.9
Correct / Wrong Hints	5.2	29.8	20.2	19.6	1.1	5.4	2.3	16.4



(a) Pre-RL / Post-RL (b) Smaller / Larger Models (c) Correct / Wrong Hints

Figure 5: A case study that the three contrasts strengthen different reasoning tokens. The shade of color represents the relative magnitude of Δ_t.

math-reasoning traces and label every token with one of eight problem-solving episodes (Schoenfeld, 2014) using ThinkARM (Li et al., 2026b), an automatic episode classifier. Table 6 shows the token distribution. Relative to the scale contrast, the post-RL and hint contrasts place more weight on the reasoning *framework* such as the *Plan* and *Monitor* episodes that structure and track the solution, whereas the scale contrast keeps more weight on the core solving steps, *Analyze* and *Implement*. The hint contrast additionally concentrates on the final *Answer* tokens, which is a consequence of conditioning on a correct versus a wrong answer. Figure 5 illustrates the same division on a single trace: **the three contrasts reinforce different kinds of tokens.** These emphases are complementary, resulting in a student learning to improve the reasoning by distilling different patterns.

5 Conclusion

We introduce W2S-OPD, a weak-to-strong on-policy distillation framework that improves a strong student using only smaller or weaker models. By isolating the capability direction between a contrast pair of weak models and re-anchoring it onto the student's base model, W2S-OPD synthesizes a proxy teacher that couples the isolated capability while staying distributionally adjacent to the student. Across math reasoning and code generation benchmarks, W2S-OPD substantially outperforms OPD, surpasses the weak teacher itself, and remains effective when the contrast pair comes from RL training, model scale, or contrastive hints, while preserving out-of-domain capabilities.

Discussion. As models approach the frontier, an ever-stronger teacher is no longer available, leaving the source of OPD supervision an open question. Weak supervision sources are abundant, yet how far they can push a stronger student before weak-to-strong supervision saturates, and how to elicit more informative signals from them, remain to be explored. Future work may develop weak-to-strong learning into a sustained paradigm for OPD and post-training more broadly, where frontier models continue to improve from supervision sources cheaper and weaker than themselves.

AI USE DISCLOSURE

We used generative AI tools solely to polish the writing of this paper. The authors reviewed and edited all AI-assisted text and take full responsibility for the content.

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.

AI-MO. AIMO validation AIME. https://huggingface.co/datasets/AI-MO/aimo-validation-aime, 2024. 90 problems from AIME 2022–2024, extracted from the AoPS wiki.

Mislav Balunovic, Jasper Dekoninck, Ivo Petrov, Nikola Jovanovic, and Martin Vechev. Matharena: Evaluating llms on uncontaminated math competitions, february 2025. *URL https://matharena. ai*, 8, 2025.

Collin Burns, Pavel Izmailov, Jan Hendrik Kirchner, Bowen Baker, Leo Gao, Leopold Aschenbrenner, Yining Chen, Adrien Ecoffet, Manas Joglekar, Jan Leike, et al. Weak-to-strong generalization: Eliciting strong capabilities with weak supervision. *arXiv preprint arXiv:2312.09390*, 2023.

Edoardo Cetin, Tianyu Zhao, and Yujin Tang. Reinforcement learning teachers of test time scaling. *Advances in Neural Information Processing Systems*, 38:107533–107567, 2026.

Alvin Chan, Yew-Soon Ong, Bill Pung, Aston Zhang, and Jie Fu. Cocon: A self-supervised approach for controlled text generation. *arXiv preprint arXiv:2006.03535*, 2020.

Tianlei Chen, Jiao Ou, Ziyuan Liu, Ruiming Tang, Jian Liang, and Han Li. Counteraction-aware multi-teacher on-policy distillation for general capability recovery with domain preservation. *arXiv preprint arXiv:2605.27115*, 2026.

Paul Christiano, Buck Shlegeris, and Dario Amodei. Supervising strong learners by amplifying weak experts. *arXiv preprint arXiv:1810.08575*, 2018.

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

Shiyuan Feng, Huan-ang Gao, Haohan Chi, Hanlin Wu, Zhilong Zhang, Zheng Jiang, Bingxiang He, Wei-Ying Ma, Ya-Qin Zhang, and Hao Zhou. Weak-to-strong generalization via direct on-policy distillation. *arXiv preprint arXiv:2607.05394*, 2026.

Scott Geng, Hamish Ivison, Chun-Liang Li, Maarten Sap, Jerry Li, Ranjay Krishna, and Pang Wei Koh. The delta learning hypothesis: Preference tuning on weak data can yield strong gains. *arXiv preprint arXiv:2507.06187*, 2025.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Minillm: Knowledge distillation of large language models. In *International Conference on Learning Representations*, volume 2024, pp. 32694–32717, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 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, de-contaminated, and verifiable mathematical dataset for advancing reasoning. *arXiv preprint arXiv:2504.11456*, 2025.

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, 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. In *International Conference on Learning Representations*, volume 2025, pp. 58791–58831, 2025.

Ijun Jang, Jewon Yeom, Juan Yeo, Hyunggyu Lim, and Taesup Kim. Stable on-policy distillation through adaptive target reformulation. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 42217–42227, 2026.

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.

Nitish Shirish Keskar, Bryan McCann, Lav R Varshney, Caiming Xiong, and Richard Socher. Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*, 2019.

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.

Ben Krause, Akhilesh Deepak Gotmare, Bryan McCann, Nitish Shirish Keskar, Shafiq Joty, Richard Socher, and Nazneen Fatema Rajani. Gedi: Generative discriminator guided sequence generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pp. 4929–4952, 2021.

Jiaze Li, Hao Yin, Haoran Xu, Boshen Xu, Wenhui Tan, Zewen He, Jianzhong Ju, Zhenbo Luo, and Jian Luan. Video-opd: Efficient post-training of multimodal large language models for temporal video grounding via on-policy distillation. *arXiv preprint arXiv:2602.02994*, 2026a.

Ming Li, Chenrui Fan, Yize Cheng, Soheil Feizi, and Tianyi Zhou. Schoenfeld's anatomy of mathematical reasoning by language models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 32773–32802, 2026b.

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*, 2026c.

Yijun Liang, Yunjie Tian, Yijiang Li, Yuqi Jia, Furong Huang, Tianyi Zhou, and Di Fu. Visual contrastive self-distillation. *arXiv preprint arXiv:2607.21556*, 2026.

Alisa Liu, Maarten Sap, Ximing Lu, Swabha Swayamdipta, Chandra Bhagavatula, Noah A Smith, and Yejin Choi. Dexperts: Decoding-time controlled text generation with experts and anti-experts. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 6691–6706, 2021.

Alisa Liu, Xiaochuang Han, Yizhong Wang, Yulia Tsvetkov, Yejin Choi, and Noah A Smith. Tuning language models by proxy. *arXiv preprint arXiv:2401.08565*, 2024.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. *Advances in neural information processing systems*, 36:21558–21572, 2023.

Ruiqi Liu, Xiaolei Lv, Gengsheng Li, Ximo Zhu, Zhiheng Wang, Zhengbo Zhang, Junkai Chen, Zhiheng Li, Bo Li, Jun Gao, et al. Visual-advantage on-policy distillation for vision-language models. *arXiv preprint arXiv:2605.21924*, 2026.

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.

Xuefei Ning, Zifu Wang, Shiyao Li, Zinan Lin, Peiran Yao, Tianyu Fu, Matthew B Blaschko, Guohao Dai, Huazhong Yang, and Yu Wang. Can llms learn by teaching for better reasoning? a preliminary study. *Advances in Neural Information Processing Systems*, 37:71188–71239, 2024.

Leyi Pan, Shuchang Tao, Yunpeng Zhai, Lingzhe Zhang, Zhaoyang Liu, Bolin Ding, Aiwei Liu, and Lijie Wen. Rlcsd: Reinforcement learning with contrastive on-policy self-distillation. *arXiv preprint arXiv:2606.11709*, 2026.

Shrimai Prabhumoye, Alan W Black, and Ruslan Salakhutdinov. Exploring controllable text generation techniques. In *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 1–14, 2020.

Valentina Pyatkin, Saumya Malik, Victoria Graf, Hamish Ivison, Shengyi Huang, Pradeep Dasigi, Nathan Lambert, and Hanna Hajishirzi. Generalizing verifiable instruction following. *Advances in Neural Information Processing Systems*, 38, 2026.

Lianhui Qin, Sean Welleck, Daniel Khashabi, and Yejin Choi. Cold decoding: Energy-based constrained text generation with langevin dynamics. *Advances in Neural Information Processing Systems*, 35:9538–9551, 2022.

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. *arXiv preprint arXiv:2311.12022*, 2023.

Alan H Schoenfeld. *Mathematical problem solving*. Elsevier, 2014.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *CoRR*, abs/2402.03300, 2024. doi: 10.48550/ARXIV.2402.03300. URL https://doi.org/10.48550/arXiv.2402.03300.

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

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

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.

Xingrun Xing, Haoqing Wang, Boyan Gao, Ziheng Li, and Yehui Tang. Trust region on-policy distillation. *arXiv preprint arXiv:2606.01249*, 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*, 2025.

Shenzhi Yang, Guangcheng Zhu, Bowen Song, Haobo Wang, Mingxuan Xia, Xing Zheng, Yingfan Ma, Zhongqi Chen, Weiqiang Wang, and Gang Chen. Oprd: On-policy representation distillation. *arXiv preprint arXiv:2606.06021*, 2026a.

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*, 2026b.

Yuqing Yang, Yan Ma, and Pengfei Liu. Weak-to-strong reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 8350–8367, 2024.

Wei Yao, Gengze Xu, Huayi Tang, Wenkai Yang, Donglin Di, Ziqiao Wang, and Yong Liu. On weak-to-strong generalization and f-divergence. *arXiv preprint arXiv:2506.03109*, 2025a.

Wei Yao, Wenkai Yang, Ziqiao Wang, Yankai Lin, and Yong Liu. Revisiting weak-to-strong generalization in theory and practice: Reverse kl vs. forward kl. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 2860–2888, 2025b.

Tianzhu Ye, Li Dong, Xun Wu, Shaohan Huang, and Furu Wei. On-policy context distillation for language models. *arXiv preprint arXiv:2602.12275*, 2026.

Hee Suk Yoon, Eunseop Yoon, Jaehyun Jang, SooHwan Eom, Ji Woo Hong, Mark Hasegawa-Johnson, Qi Dai, Chong Luo, and Chang D Yoo. Decomposed on-policy distillation for vision-language reasoning: Steering gradients for visual grounding. *arXiv preprint arXiv:2606.00564*, 2026.

Fangxu Yu, Lai Jiang, Haoqiang Kang, Shibo Hao, and Lianhui Qin. Flow of reasoning: Training llms for divergent reasoning with minimal examples. *arXiv preprint arXiv:2406.05673*, 2024.

Qianhao Yuan, Jie Lou, Xing Yu, Hongyu Lin, Le Sun, Xianpei Han, and Yaojie Lu. Vision-opd: Learning to see fine details for multimodal llms via on-policy self-distillation. *arXiv preprint arXiv:2605.18740*, 2026a.

Yige Yuan, Teng Xiao, Shuchang Tao, Xue Wang, Jinyang Gao, Bolin Ding, and Bingbing Xu. Incentivizing strong reasoning from weak supervision. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7138–7156, 2026b.

Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, et al. Glm-5: from vibe coding to agentic engineering. *arXiv preprint arXiv:2602.15763*, 2026.

Yifan Zhang and Team Math-AI. American invitational mathematics examination (aime) 2025, 2025.

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.

Xuandong Zhao, Xianjun Yang, Tianyu Pang, Chao Du, Lei Li, Yu-Xiang Wang, and William Yang Wang. Weak-to-strong jailbreaking on large language models. *arXiv preprint arXiv:2401.17256*, 2024.

Wenhong Zhu, Ruobing Xie, Rui Wang, and Pengfei Liu. Hybrid policy distillation for llms. *arXiv preprint arXiv:2604.20244*, 2026.

A Related Work

On-Policy Distillation. OPD (Lu & Lab, 2025; Song & Zheng, 2026; Agarwal et al., 2024) supervises the student on its own rollouts with a superior teacher. It has become a key post-training paradigm for strong-to-weak distillation (Zeng et al., 2026; Yang et al., 2025) and for merging multi-domain experts into one model (Xiao et al., 2026; Xu et al., 2026; Chen et al., 2026; Yang et al., 2026a), with follow-ups relaxing its requirements through self-distillation from privileged information (Zhao et al., 2026; Shenfeld et al., 2026; Hübotter et al., 2026; Ye et al., 2026), stabilized optimization against the student–teacher gap (Jin et al., 2026; Ko et al., 2026; Xing et al., 2026; Jang et al., 2026), and multimodal extensions (Li et al., 2026a; Liu et al., 2026; Yoon et al., 2026; Yuan et al., 2026a; Liang et al., 2026). These works assume a teacher at least as capable as the student. In contrast, W2S-OPD works weak-to-strong, synthesizing a teacher from weak models rather than requiring a strong superior model. Direct-OPD (Feng et al., 2026) is a concurrent work that transfers a weak teacher's pre-/post-RL log-ratio as a dense reward. However, its contrast is confined to RL-trained teachers, without incorporating directions from multiple contrastive sources, and is only evaluated on math reasoning tasks.

Weak-to-Strong Generalization. Weak-to-Strong elicits the capabilities of a stronger model with the supervision of a weak model (Burns et al., 2023). This is critical when a stronger model is hard to obtain (Christiano et al., 2018). Recent work extends it to LLM reasoning and alignment (Yao et al., 2025b;a; Yuan et al., 2026b; Yang et al., 2024; Zhao et al., 2024; Geng et al., 2025). Another line casts learning by teaching, where a strong model improves by instructing weaker students and turning their comprehension into a training signal (Ning et al., 2024; Cetin et al., 2026). However, such supervision primarily provides trajectory-level signals, while W2S-OPD instead distills a synthesized proxy teacher with token-level dense supervision.

Controllable Text Generation. Controlling the outputs of an LLM has been widely studied (Prabhumoye et al., 2020) along two directions: training-based methods finetune the model to elicit desired properties (Keskar et al., 2019; Chan et al., 2020) but are computationally costly, whereas decoding-time methods steer generation by shifting or composing output distributions at inference (Liu et al., 2021; Krause et al., 2021; Qin et al., 2022). W2S-OPD adopts this decoding-time composition to synthesize its proxy teacher, but turns it into a training signal that transfers the ability from weak models to a strong student.

B KL-Divergence Estimation

Single-token OPD. An alternative estimator, adopted by several recent OPD implementations (Lu & Lab, 2025; Zhu et al., 2026), extracts a single scalar of teacher information per sampled token: the log-ratio on the sampled token is treated as a token-level advantage,

$$A_t \;=\; \log \pi_T(y_t \mid s_t) \;-\; \log \pi_S^{\text{old}}(y_t \mid s_t), \tag{6}$$

and plugged into a clipped policy-gradient update, where π_S^{old} denotes the policy that generated the rollouts. This variant reuses the RL infrastructure, but conveys only 1 scalar per position and thus exhibits higher variance than the dense top-k objective, which backpropagates the teacher's distribution at every position through a directly differentiable divergence and requires no advantage estimation or clipping. We therefore adopt the dense top-k objective for both distillation arms in all experiments.

Top-k OPD. Materializing the teacher's full distribution requires storing and transferring $|\mathcal{V}|$ logits (roughly 150K for the Qwen3 family) at every response position, which is prohibitively expensive to communicate across training workers. Since the teacher distribution is highly peaked, its top-k tokens already carry the majority of probability mass at nearly all positions. The teacher worker therefore ships only the top-k token indices and log-probabilities per position, and the divergence in Eq. 1 is estimated on this truncated support:

$$\mathcal{L}_{\text{OPD}}^{\text{top-}K}(\pi_S) \;=\; \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_S(\cdot|x)}\left[\sum_{t=1}^{|y|} \mathrm{KL}\big(\bar{\pi}_S^{\mathcal{K}_t} \,\big\|\, \bar{\pi}_T^{\mathcal{K}_t}\big)\right], \tag{7}$$

where $\mathcal{K}_t$ denotes the top-k support of the teacher distribution at position t, and $\bar{\pi}^{\mathcal{K}_t}$ denotes the corresponding distribution restricted to $\mathcal{K}_t$ and renormalized. The truncation error is bounded by

the tiny probability mass that falls outside the teacher's top-k set, so the estimation preserves the supervision almost losslessly while reducing the cost.

C From logit arithmetic to a KL-constrained objective

Let $r(s,a) := \log \pi^+(a \mid s) - \log \pi^-(a \mid s)$ denote the *capability reward* carried by the contrast pair: a token-level, relative quantity measuring how much more the positive model favors a than the negative one does, in which the absolute competence of either source never appears. Since $e^{z^+(a)-z^-(a)}$ equals $\pi^+(a \mid s)/\pi^-(a \mid s)$ up to a factor independent of a, which the softmax normalizer absorbs, Eq. 2 is an exponential tilting of the student's own base model,

$$\pi_{T,\alpha}(a \mid s) = \frac{\pi_{\text{base}}(a \mid s)\, e^{\alpha r(s,a)}}{Z_\alpha(s)}, \qquad Z_\alpha(s) = \mathbb{E}_{a\sim\pi_{\text{base}}(\cdot|s)}\big[e^{\alpha r(s,a)}\big]. \tag{8}$$

Substituting Eq. 8 into $\mathrm{KL}(q \,\|\, \pi_{T,\alpha})$ and rearranging gives, for every $q \in \Delta(\mathcal{V})$,

$$\underbrace{\mathbb{E}_{a\sim q}\big[r(s,a)\big] - \tfrac{1}{\alpha}\mathrm{KL}\big(q \,\|\, \pi_{\text{base}}(\cdot \mid s)\big)}_{\mathcal{J}_\alpha(q\,;\,s)} \;=\; \tfrac{1}{\alpha}\log Z_\alpha(s) \;-\; \tfrac{1}{\alpha}\mathrm{KL}\big(q \,\|\, \pi_{T,\alpha}(\cdot \mid s)\big). \tag{9}$$

The first term on the right does not depend on q and the second is nonnegative and vanishes only at $q = \pi_{T,\alpha}$. The proxy teacher is therefore the unique closed-form maximizer

$$\pi_{T,\alpha}(\cdot \mid s) \;=\; \arg\max_{q\in\Delta(\mathcal{V})} \Big\{ \underbrace{\mathbb{E}_{a\sim q}\big[r(s,a)\big]}_{\text{inject capability}} \;-\; \tfrac{1}{\alpha}\underbrace{\mathrm{KL}\big(q \,\|\, \pi_{\text{base}}(\cdot \mid s)\big)}_{\text{stay adjacent to the student}} \Big\}, \tag{10}$$

where $1/\alpha$ is the strength of a trust region centered at the student rather than at any weak model.

A compositional view. Eq. 8 can be stated without reference to logits at all,

$$\pi_{T,\alpha}(a \mid s) \;\propto\; \pi_{\text{base}}(a \mid s) \left(\frac{\pi^+(a \mid s)}{\pi^-(a \mid s)}\right)^{\alpha}, \tag{11}$$

which exhibits the proxy teacher as a product of experts over the shared vocabulary: the student's base model is the anchor, π^+ enters as a conjoined expert and π^- as a negated one, and α is the exponent controlling how sharply the pair is applied. The capability direction is thus a *negation* operation, which divides out what the two weak models agree on, and the trust region in Eq. 10 is precisely the price of keeping the resulting product near the anchor, with $\frac{1}{\alpha}\log Z_\alpha(s)$ its free energy at s. With K contrast pairs the same reading gives:

$$\pi_{T,\boldsymbol{\alpha}}(a \mid s) \;\propto\; \pi_{\text{base}}(a \mid s) \prod_{k=1}^{K} \left(\frac{\pi_k^+(a \mid s)}{\pi_k^-(a \mid s)}\right)^{\alpha_k}, \tag{12}$$

so merging capability directions is multiplication of experts. In particular the result is invariant to the order in which the pairs are introduced, unlike sequentially fine-tuning or distilling one teacher after another, where the outcome depends on the schedule and earlier skills may be overwritten.

Three design choices follow: i) *Adjacency*: the reference measure in Eq. 10 is the student's own base model, so the proxy teacher is a reweighting of a distribution the student already realizes, and α sets how far it may move; $\alpha \to 0$ recovers π_{base} and supplies no supervision, whereas $\alpha \to \infty$ removes the trust region and collapses onto $\arg\max_a r(s,a)$, so a moderate α is optimal (Figure 4). ii) *Contrast*: the weak models enter only through the ratio r, so a pair with no capability gap leaves the anchor untouched and the student is never ceilinged at the sources, unlike direct OPD whose optimum is π^+ itself. iii) *Composition*: rewards add where policies do not, so Eq. 4 is the same maximizer with composite reward $\sum_k \alpha_k r_k$, and K capability directions are merged by a single distillation run.

D Implementation Details

All experiments are implemented on top of the verl framework, and each distillation run is conducted on 2 NVIDIA B200 GPUs. Table 7 lists the training hyperparameters shared by W2S-OPD and OPD. For the proxy teacher, the divergence is estimated on the teacher's top-k support with $K = 32$ (Appendix B). Table 8 reports the GRPO recipe used to obtain domain experts in the pre-RL / post-RL setting, and Table 9 shows the rollout instructions used during OPD training.

Table 7: Training hyperparameters of W2S-OPD. Paired entries denote math / code.

Hyperparameter	Value
Algorithm	On-Policy Distillation
Student Model	Qwen3-8B
Positive / Negative	Qwen3-4B-RL / Qwen3-4B
Amplification Coeff. (α)	1.0 / 0.75
Teacher Top-k	32
Max Prompt Length	2048
Max Response Length	16384
Learning Rate	2e-6
LR Scheduler	Constant
Sampling Temp. / Top-p	1.0 / 1.0
Batch Size	64
Training Steps	100

Table 8: Training hyperparameters of the domain experts (RL-trained positive models).

Hyperparameter	Value
Algorithm	GRPO
Base Models	Qwen3-4B
Reasoning Mode	Non-Thinking
Max Prompt Length	2048
Max Response Length	16384
KL Loss Coefficient	0
Learning Rate	1e-6
LR Scheduler	Constant
Sampling Temp. / Top-p	1.0 / 1.0
Batch Size	128
Mini-batch Size	128
Rollout Number (G)	8
Training Steps	500 / 300
Math Data	DeepMath-103K (L6)
Code Data	Eurus-RL-Code

Table 9: On-policy rollout prompts used during OPD training.

Domain	Prompt
Math	`{problem}\n Please reason step by step, and put your final answer within \boxed{}.`
Code	`{problem}\n Write Python code to solve the problem. Present the code in \n“‘python\n Your code\n“‘\n at the end. You need to think first then write the Python code.`

E Additional Experimental Results

E.1 Runtime Analysis

Training efficiency matters for practical deployment. To quantify the overhead W2S-OPD introduces, we record the average wall-clock time per training step in the pre-RL / post-RL setting and report it in Table 10. W2S-OPD incurs only a 20% increase in per-step time over OPD, despite forwarding 3 frozen models (the 8B anchor and the 4B contrast pair) instead of a single 4B teacher. The overhead stays modest because the per-step cost is dominated by on-policy rollout generation, which the two methods share, while the additional teacher-side scoring is forward-only. These scoring passes are also independent of one another and can be further parallelized across devices.

Table 10: Average wall-clock time per training step (s) in the pre-RL / post-RL setting; W2S-OPD adds only 20% over OPD.

Method	Training time per step (s)
OPD	868
W2S-OPD	1043

E.2 Performance with Different Scales of RL Experts

To examine how the scale of the RL expert affects W2S-OPD, we replace the 4B expert with Qwen3-0.6B and Qwen3-1.7B experts, each obtained by the identical GRPO recipe, and distill the same Qwen3-8B student. As shown in Table 11, W2S-OPD improves the student and outperforms OPD under every expert scale. Even the 0.6B expert, more than 13$\times$ smaller than the student, still improves it, confirming that the direction of the RL-induced capacity is transferable on every scale.

E.3 Performance with Different Scales of Base Models

To investigate how the choice of base models affects the Smaller / Larger setting, we fix the Qwen3-8B student and the Qwen3-4B positive model, and vary the negative model between Qwen3-0.6B and Qwen3-1.7B. As shown in Table 12, both pairs improve the student, and the

Table 11: Results for Pre-RL / Post-RL contrast setting with different teacher scales on math reasoning. *Improv.* reports the absolute gain of W2S-OPD over OPD.

Method	AIME24	AIME25	HMMT25 (Feb.)	HMMT25 (Nov.)	Avg.
Student Base (8B)	26.1	21.1	11.6	9.0	17.0
0.6B Expert					
Neg. Model (0.6B)	1.6	2.3	0.1	2.7	1.7
Pos. Model (0.6B-RL)	13.5	21.2	10.7	7.0	13.1
OPD	31.9	31.4	17.0	14.8	23.7
W2S-OPD	**54.4**	**44.5**	**24.4**	**32.5**	**39.0**
Improv.	**+22.5**	**+13.1**	**+7.4**	**+17.7**	**+15.3**
1.7B Expert					
Neg. Model (1.7B)	11.6	12.6	6.0	3.6	8.5
Pos. Model (1.7B-RL)	40.0	32.0	19.6	18.3	27.5
OPD	42.7	35.5	19.0	20.0	29.3
W2S-OPD	**50.3**	**38.5**	**19.6**	**28.3**	**34.2**
Improv.	**+7.6**	**+3.0**	**+0.6**	**+8.3**	**+4.9**

Table 13: Results for Pre-RL / Post-RL contrast setting with larger-scale student model on math reasoning. *Improv.* reports the absolute gain of W2S-OPD over OPD.

Method	AIME24	AIME25	HMMT25 (Feb.)	HMMT25 (Nov.)	Avg.
Neg. Model (4B)	21.6	19.9	10.0	8.2	14.9
Pos. Model (4B-RL)	62.0	57.9	34.8	40.4	48.8
Student Base (30B)	28.7	21.6	11.0	12.1	18.3
OPD	62.8	53.5	33.0	39.5	47.2
W2S-OPD	**67.0**	**57.7**	**36.2**	**42.9**	**50.9**
Improv.	**+4.2**	**+4.2**	**+3.2**	**+3.4**	**+3.7**

gain grows with the capability gap within the pair: the wider 4B−0.6B pair yields the larger improvement of 6.0 points on math reasoning, whereas the narrower 4B−1.7B pair yields 4.7 points. We attribute this to the amount of capability the subtraction isolates, where a larger gap spans a longer stretch of the direction along which capability grows with scale, so the difference extracts a stronger and more informative signal, and re-anchoring it onto the student base provides supervision that pushes the student further. These results suggest instantiating W2S-OPD with the most separated base-model pair available.

Table 12: Average math reasoning accuracy of W2S-OPD with base-model contrast pairs of different capability gaps. The student is Qwen3-8B.

Contrast Pair	4B−0.6B	4B−1.7B
W2S-OPD	23.0	21.7
Improv. over Student	**+6.0**	**+4.7**

E.4 Scale up to Larger MoE LLMs

To examine whether the capability direction extracted from a small dense contrast pair remains transferable to a sparsely activated architecture, we scale the student from Qwen3-8B to Qwen3-30B-A3B, an MoE model with 30B total and 3B activated parameters, while keeping the pre-RL / post-RL contrast pair and every training hyperparameter unchanged. As shown in Table 13, W2S-OPD transfers to the MoE student without any modification, reaching 50.9% average accuracy on math reasoning and outperforming OPD by 3.7 points on average on all four benchmarks. Notably, W2S-OPD again surpasses the 4B-RL expert that supplies the capability direction (50.9 versus 48.8), whereas OPD, which imitates that expert directly, remains below it at 47.2, confirming the capability ceiling argument at a new scale. The experimental results show that W2S-OPD can be scaled up to LLMs with larger-scale parameters and has the potential to be applied to more real-world scenarios.

F EXAMPLE GENERATIONS

We provide qualitative examples shown in Figures 6–9. Figures 6 and 7 come from the pre-RL / post-RL setting and compare W2S-OPD against OPD; Figures 8 and 9 come from the smaller/larger setting and compare the Qwen3-8B student before and after W2S-OPD distillation. In each case the W2S-OPD student succeeds while the baseline fails, with the baseline's errors marked in red.

Case Study: Math Reasoning (Pre-RL / Post-RL Setting)

Question: Let p be the least prime number for which there exists a positive integer n such that n^4+1 is divisible by p^2. Find the least positive integer m such that m^4+1 is divisible by p^2.
Correct Answer: 110

Direct OPD:
We need to find the least prime p such that there exists a positive integer n with n^4+1 divisible by p^2. Then, once we find that p, we need to find the smallest positive integer m such that m^4+1 is divisible by p^2.
Step 1: Understand the problem. We are looking for the smallest prime p such that there exists an integer n where $n^4+1 \equiv 0 \pmod{p^2}$, *i.e.,* $n^4 \equiv -1 \pmod{p^2}$. Then, once we find such p, we need to find the minimal positive integer m satisfying the same congruence.
Step 2: Find the minimal prime p. To solve this congruence, we need to know when -1 is a quartic residue. Let's try small primes:
Try $p=2$: we need n^4+1 divisible by 4. $n=1$: 2; $n=2$: 17; $n=3$: 82; $n=4$: $257 \equiv 1 \pmod 4$ — none divisible by 4. No solution.
Try $p=3$: we need $n^4 \equiv -1 \equiv 8 \pmod 9$. Checking $n=0,\dots,8$: $0,1,7,0,4,4,0,7,1$ (*e.g.,* $5^2=25\equiv 7$, so $5^4 \equiv 7^2 = 49 \equiv 4$; $8 \equiv -1$, so $8^4 \equiv 1$). None give 8. No solution.
Try $p=5$: check mod 5 first: $x^4 \bmod 5$ for $x=0,\dots,4$ gives $0,1,1,1,1$, so all $x^4 \equiv 0$ or $1 \pmod 5$, and $-1 \equiv 4$ is not a quartic residue. Not possible.
Try $p=7$: $x^4 \bmod 7$ takes values $\{0,1,2,4\}$ ($2^4=16\equiv 2$; $3^4=81\equiv 4$; $5^4\equiv 2$; $6\equiv -1$ gives 1). $-1\equiv 6$ is not among them. No solution.
Try $p=11$: computing x^2 then $x^4 \bmod 11$ for $x=1,\dots,10$ gives quartic residues $\{1,3,4,5,9\}$; $-1\equiv 10$ is not among them. No solution.
Try $p=13$: $-1\equiv 12$. Checking: $3^4\equiv 3$; $4^4\equiv 9$; $5^2=25\equiv 12$, $5^4\equiv 12^2=144\equiv 1$; $6^4\equiv 9$; $7^4\equiv 9$; $8^2=64\equiv 12$, $8^4\equiv 1$; $9^4\equiv 9$; $10^4\equiv 3$; $11^4\equiv 3$; $12^4\equiv 1$. No x with $x^4\equiv 12$. No solution.
Maybe we can think about orders: in the cyclic group mod p, the equation $x^4\equiv -1$ has solutions iff -1 lies in the image of the 4th-power map…this is getting complex. Let me try concrete examples.
Try $p=17$: try $x=2$: $2^4=16\equiv -1 \pmod{17}$. Yes! So $2^4\equiv -1 \pmod{17}$, and $p=17$ is such a prime. Since we checked $p=2,3,5,7,11,13$ and found no solutions, and there is no prime between 13 and 17, $p=17$ is the smallest such prime.
Step 3: Find the least m with $m^4+1\equiv 0 \pmod{289}$. We know $2^4\equiv -1 \pmod{17}$, but $2^4+1=17$ is divisible by 17, not by 17^2. So we lift with Hensel's lemma. Write $m=2+17k$; modulo 17^2, terms with $(17k)^2$ vanish, so

$$(2+17k)^4 \equiv 2^4 + 4\cdot 2^3\cdot(17k) = 16+32\cdot 17k \pmod{289}.$$

We want $16+544k\equiv -1 \pmod{289}$. Since $544\equiv 255 \pmod{289}$, this is $255k\equiv -17 \pmod{289}$, and $255\equiv -34$, so $-34k\equiv -17$, *i.e.,* $34k\equiv 17 \pmod{289}$. Dividing by 17: $2k\equiv 1 \pmod{17}$, so $k\equiv 9 \pmod{17}$ (since $2\times 9=18\equiv 1$). Therefore $m=2+17\times 9=155$.
Is there a smaller m? If we consider $m\equiv 2 \pmod{17}$, the possible m's are $2, 19, 36, 53, 70, 87, 104, 121, 138, 155, \dots$ Check $m=2$: $2^4+1=17$, not divisible by 289. Check $m=19$: $19^2=361\equiv 72 \pmod{289}$, $19^4\equiv 72^2=5184\equiv 271 \pmod{289}$ (since $289\times 17=4913$ and $5184-4913=271$), so $19^4+1\equiv 272\neq 0$. Not divisible. Checking $m=36$ would be tedious. Alternatively, note that the solution is unique in the congruence class, so the minimal m is 155. Since we used Hensel's lemma and the lifting worked, we can trust that this is the correct answer. Thus, the least positive integer m is 155.
Final Answer: $\boxed{155}$ (← the lift itself is valid, $155^4\equiv -1 \bmod 289$, but the trace only examines the root $m\equiv 2 \pmod{17}$; the congruence $x^4\equiv -1 \pmod{17}$ has 4 roots (2, 8, 9, 15), and lifting the root 8 gives $110<155$.)

W2S-OPD:

We are given a problem involving prime numbers and divisibility. Let me break it down step by step.
Step 1: Understand the problem. We need two things: (1) find the smallest prime p such that p^2 divides $n^4 + 1$ for some positive integer n; (2) for that p, find the smallest m such that p^2 divides $m^4 + 1$. So first, we need to determine what primes p satisfy the condition that there exists some n where $n^4 \equiv -1 \pmod{p^2}$.
Step 2: Analyze the equation $n^4 \equiv -1 \pmod{p^2}$. This is equivalent to $n^8 \equiv 1 \pmod{p^2}$, since squaring both sides gives $(n^4)^2 \equiv (-1)^2 = 1 \pmod{p^2}$. So the order of n modulo p^2 must divide 8, but not divide 4 (since $n^4 \equiv -1 \neq 1$). Let me start checking small primes and see which ones satisfy this condition.
Step 3: Check small primes. For $p = 2$: $n = 1$ gives $1 + 1 = 2$, not divisible by $2^2 = 4$; $n = 2$ gives $16 + 1 = 17$, not divisible by 4. No solution. For $p = 3$: $n = 1, \ldots, 6$ give $n^4 + 1 \equiv 2, 8, 1, 5, 5, 1 \pmod{9}$, none zero. No solution. For $p = 5$: checking $n = 1, \ldots, 10$ against $p^2 = 25$, the values $n^4 + 1 \bmod 25$ are $2, 17, 7, 7, 1, 22, 2, 22, 12, 1$, none zero. No solution.
For larger primes, let me think algebraically. If p divides $n^4 + 1$, then $n^4 \equiv -1 \pmod{p}$, hence $n^8 \equiv 1 \pmod{p}$, so the order of n modulo p is exactly 8. Since the multiplicative group mod p is cyclic of order $p - 1$, we need $8 \mid p - 1$, *i.e.,* $p \equiv 1 \pmod{8}$. This rules out $p = 7, 13$ (verified directly: no n with $n^4 \equiv -1$ exists for either). The smallest prime with $p \equiv 1 \pmod{8}$ is $p = 17$.
Step 4: Verify $p = 17$. The group mod 17 has order 16, so elements of order 8 exist. Computing $n^4 \bmod 17$ for small n: at $n = 8$, $8^2 = 64 \equiv 13$, so $8^4 \equiv 13^2 = 169 \equiv -1 \pmod{17}$. So $n = 8$ satisfies $n^4 + 1 \equiv 0 \pmod{17}$.
But we need $p^2 \mid n^4 + 1$. For $n = 8$: $8^4 + 1 = 4097$, and $4097 = 14 \times 289 + 51$, so $4097 \equiv 51 \pmod{289}$ — divisible by 17 but not by 17^2. So we must lift the solution.
Step 5: Hensel lifting. Write $n = 8 + 17k$ and expand modulo 17^2, dropping terms with $(17k)^2$:

$$(8 + 17k)^4 + 1 \equiv 8^4 + 4 \cdot 8^3 \cdot 17k + 1 \pmod{289}.$$

Since $8^4 = 4096 \equiv 50 \pmod{289}$, the condition becomes $51 + 4 \cdot 8^3 \cdot 17k \equiv 0 \pmod{289}$. Note $51 = 3 \times 17$, so dividing through by 17 reduces the condition to

$$4 \cdot 8^3 \cdot k \equiv -3 \pmod{17}.$$

Compute $8^2 \equiv 13$, $8^3 \equiv 8 \cdot 13 = 104 \equiv 2 \pmod{17}$, so $4 \cdot 8^3 \equiv 8 \pmod{17}$ and the condition is $8k \equiv -3 \equiv 14 \pmod{17}$. The inverse of 8 mod 17 is 15 (since $8 \times 15 = 120 \equiv 1$), giving $k \equiv 14 \times 15 = 210 \equiv 6 \pmod{17}$. Therefore $n = 8 + 17 \times 6 = 110$ satisfies $n^4 + 1 \equiv 0 \pmod{289}$.
Step 6: Check that no smaller m works. Any solution must satisfy $m^4 \equiv -1 \pmod{17}$, so $m \equiv 8$ or $m \equiv 9 \pmod{17}$ (among the order-8 residues). For the root $m \equiv 9 \pmod{17}$: $9^4 = 6561 \equiv 203 \pmod{289}$, and repeating the lifting with $9^3 \equiv 15$, $4 \cdot 9^3 \equiv 9 \pmod{17}$ gives $9k \equiv -12 \equiv 5 \pmod{17}$; the inverse of 9 is 2, so $k \equiv 10 \pmod{17}$ and the minimal solution on this branch is $m = 9 + 170 = 179 > 110$.
For the branch $m \equiv 8 \pmod{17}$, the candidates below 110 are $8, 25, 42, 59, 76, 93$, corresponding to $k = 0, 1, \ldots, 5$; since the lifting requires $k \equiv 6 \pmod{17}$, none of them works (checked explicitly for $m = 8$: $4097 \equiv 51$; $m = 25$: $25^2 = 625 \equiv 47$, $47^2 = 2209 \equiv 186$, $186 + 1 = 187 \neq 0 \pmod{289}$). Therefore $m = 110$ is the least positive integer with $17^2 \mid m^4 + 1$.
Final Answer: $\boxed{110}$

Figure 6: Math case study under Pre-RL/ Post-RL setting, with the errors of direct OPD marked in red.

Case Study: Code Generation (Pre-RL/Post-RL Setting)

Question: Write a Python function to find the last digit of a given number.

Direct OPD:
To find the **last digit** of a given number, we can use the **modulo 10** operation.
Reasoning: For any number `n`, the last digit is the remainder when `n` is divided by 10. For example: `123 % 10 = 3` → last digit is 3; `456 % 10 = 6` → last digit is 6; `0 % 10 = 0` → last digit is 0; `-123 % 10 = 7` → because -123 is equivalent to $-123 + 130 = 7$ (in mod 10).
But wait! When the number is negative, the modulo operation in Python returns a non-negative result. So for negative numbers, the last digit is still the correct one.
So, the function can be written as:

```
def last_Digit(n):
    return n % 10
```

Test Case: `last_Digit(123)` → `3`; `last_Digit(456)` → `6`; `last_Digit(-123)` → `7` (because `-123 % 10` is 7); `last_Digit(0)` → `0`. So this function works for all integer inputs.

Final Code:

```
def last_Digit(n):
    return n % 10
# Test the function
assert last_Digit(123) == 3
```

(← the intended last digit of −123 is 3, not 7; the trace convinces itself that Python's non-negative modulo is already the desired behavior and never takes the absolute value.)

W2S-OPD:

Let's first **think** about the problem:

Problem: We want a function that returns the **last digit** of a given number. The last digit of a number is simply the remainder when the number is divided by 10. For example: `123 % 10` gives `3`; `4567 % 10` gives `7`; `0 % 10` gives `0`.

So, the strategy is straightforward: take the given number `n` and return `n % 10`.

But we must consider the possibility of **negative numbers**. For instance: `-123 % 10` gives `7` in Python — this is not the intended behavior for this problem. We want the **absolute value's last digit**. So, we should take the absolute value of `n` before taking the modulo 10.

So the correct formula is: last digit = abs(n) % 10.

Let's test with sample input: `last_Digit(123)` → `abs(123) % 10` → `3` ✓; `last_Digit(-456)` → `abs(-456) % 10` → `6` ✓; `last_Digit(0)` → `0 % 10` → `0` ✓; `last_Digit(9)` → `9 % 10` → `9` ✓.

Now let's write the Python function.

Python Code:

```
def last_Digit(n):
    return abs(n) % 10
```

Now, test it with the assert statement:

```
assert last_Digit(123) == 3
```

This should pass without error.

Let me present the final code as you requested.

✓ **Final Answer:**

```
def last_Digit(n):
    return abs(n) % 10
```

Figure 7: Code case study under the pre-RL/post-RL setting with the errors of direct OPD marked in red.

Case Study: Math Reasoning (Smaller & Larger Setting)

Question: Let $ABCD$ and $CEFG$ be squares such that C lies on segment $\overline{DG}$ and E lies on segment $\overline{BC}$. Let O be the circumcenter of triangle AEG. Given that A, D, and O are collinear and $AB = 1$, compute FG.

Correct Answer: $\sqrt{3} - 1$

Qwen3-8B (student):

We are given two squares, $ABCD$ and $CEFG$, such that $AB = 1$; C lies on segment $\overline{DG}$; E lies on segment $\overline{BC}$; O is the **circumcenter** of triangle AEG; and A, D, and O are **collinear**. We are to find the length of FG.

Step 1: Assign coordinates. Place square $ABCD$ with $A = (0, 0)$, $B = (1, 0)$, $C = (1, 1)$, $D = (0, 1)$, so square $ABCD$ has side length 1. Since E lies on $\overline{BC}$, a vertical segment from $(1, 1)$ to $(1, 0)$, let $E = (1, t)$ with $0 \leq t \leq 1$. The direction from C to E is downward, so the square $CEFG$ extends down and left from point C. The vector from C to E is $(0, t - 1)$, and the vector from E to F should be orthogonal of the same length, so rotate the vector $(0, t-1)$ $90°$ clockwise: the rotation of (x, y) clockwise is $(y, -x)$, so the vector from E to F is $(t - 1, 0)$, giving $F = (1 + t - 1,\ t) = (t, t)$. Then from F,

move by the same vector from C to E: $G = F + (0, t-1) = (t,\ 2t-1)$. Therefore the square $CEFG$ has vertices $C = (1,1)$, $E = (1,t)$, $F = (t,t)$, $G = (t, 2t-1)$.
Step 2: Triangle AEG. We have $A = (0,0)$, $E = (1,t)$, $G = (t, 2t-1)$. We will find the circumcenter O and then enforce that A, D, O are collinear.
Step 3: Find the circumcenter of triangle AEG. Using perpendicular bisectors. For side AE: midpoint $M_1 = (\frac{1}{2}, \frac{t}{2})$, slope t, so the perpendicular bisector is

$$y = -\frac{1}{t}x + \frac{1+t^2}{2t}. \tag{1}$$

For side AG: midpoint $M_2 = (\frac{t}{2}, \frac{2t-1}{2})$, slope $\frac{2t-1}{t}$, perpendicular slope $-\frac{t}{2t-1}$, so

$$y = -\frac{t}{2t-1}x + \frac{t^2}{2t-1} + \frac{2t-1}{2} = -\frac{t}{2t-1}x + \frac{6t^2-4t+1}{2(2t-1)}. \tag{2}$$

Solving (1) and (2) directly is complicated, but we will use a key condition: A, D, and O are collinear. Since $A = (0,0)$ and $D = (0,1)$, line AD is vertical, *i.e.,* $x = 0$. So the x-coordinate of O must be 0, and from (1): $O = \left(0, \frac{1+t^2}{2t}\right)$.
Step 4: Enforce that O lies on both perpendicular bisectors. Plugging $x = 0$ into (2) gives $y = \frac{6t^2-4t+1}{2(2t-1)}$, so for consistency we require

$$\frac{1+t^2}{2t} = \frac{6t^2-4t+1}{2(2t-1)} \implies (1+t^2)(2t-1) = t(6t^2-4t+1).$$

Expanding: $2t^3 - t^2 + 2t - 1 = 6t^3 - 4t^2 + t$, which gives the cubic

$$-4t^3 + 3t^2 + t - 1 = 0.$$

Try rational roots. $t = 1$: $-4+3+1-1 = -1 \neq 0$. $t = \frac{1}{2}$: $-\frac{1}{2} + \frac{3}{4} + \frac{1}{2} - 1 = -\frac{1}{4} \neq 0$. $t = \frac{1}{4}$: $-\frac{1}{16} + \frac{3}{16} + \frac{4}{16} - \frac{16}{16} = -\frac{11}{16} \neq 0$. $t = \frac{2}{3}$: messy. Let's try an interval method: $t = 1$ gives -1; $t = \frac{3}{2}$: messy, but try a better approach.
Wait, this is very algebra-heavy. But we already found that $O = \left(0, \frac{1+t^2}{2t}\right)$, and from the earlier step, when $t = \frac{1}{2}$, this works? Plugging $t = \frac{1}{2}$: $\frac{1+1/4}{1} = \frac{5}{4}$, the y-coordinate of O; also, from tentative earlier steps, this works. So let's assume $t = \frac{1}{2}$.
Step 5: Compute FG with $t = \frac{1}{2}$. Recall $F = (t,t) = (\frac{1}{2}, \frac{1}{2})$ and $G = (t, 2t-1) = (\frac{1}{2}, 0)$. So segment FG is vertical with length $|\frac{1}{2} - 0| = \frac{1}{2}$.
Final Answer: $\boxed{\frac{1}{2}}$ (← two failures: the square is built by rotating clockwise, placing G at $(t, 2t-1)$, for which C never lies on $\overline{DG}$, and this constraint is never checked; then, unable to solve the resulting cubic, the trace assumes $t = \frac{1}{2}$ immediately after computing that $t = \frac{1}{2}$ does not satisfy the equation.)

W2S-OPD:
We are given two squares: $ABCD$ and $CEFG$, with the following constraints: point C lies on segment $\overline{DG}$; point E lies on segment $\overline{BC}$; O is the **circumcenter** of triangle AEG; points A, D, and O are **collinear**; $AB = 1$. We are to compute the length FG. We'll solve this **step by step** using coordinate geometry.
Step 1: Set up a coordinate system. Let's place square $ABCD$ in the coordinate plane for simplicity. Since $AB = 1$ and $ABCD$ is a square, we can assign $A = (0,0)$, $B = (1,0)$, $C = (1,1)$, $D = (0,1)$. So square $ABCD$ is in the first quadrant, with sides aligned with the axes.
Step 2: Place square $CEFG$. We need to place square $CEFG$ such that $C = (1,1)$, E lies on $\overline{BC}$, C lies on $\overline{DG}$, and O is the circumcenter of AEG with A, D, O collinear. Let's try to find the coordinates of E, F, and G.
Step 3: Place point E on $\overline{BC}$. Segment $\overline{BC}$ goes from $B = (1,0)$ to $C = (1,1)$, so it is a **vertical segment** at $x = 1$, from $y = 0$ to $y = 1$. Let's say $E = (1,e)$, where $0 < e < 1$.
Step 4: Place square $CEFG$. Side $\overrightarrow{CE}$ is **vertical**, going from $C = (1,1)$ to $E = (1,e)$, so it is of length $|1-e|$. The square can be constructed in two possible orientations, either to the left or to the right of CE, and we must choose the one consistent with the constraint that C lies on $\overline{DG}$. Try **counterclockwise** first: $\vec{CE} = (0, e-1)$; rotating $\vec{CE}$ 90° counterclockwise gives $(1-e, 0)$; so from $E = (1,e)$, moving in direction $(1-e, 0)$ gives $F = (2-e,\ e)$; then from F, moving opposite to $\vec{CE}$ by $(0, 1-e)$ gives $G = (2-e,\ 1)$.

Let's verify this shape is a square: $\vec{CE} = (0, e-1)$, $\vec{EF} = (1-e, 0)$, $\vec{FG} = (0, 1-e)$, $\vec{GC} = (e-1, 0)$. All sides are of length $|e-1|$, and adjacent sides are perpendicular $\rightarrow$ this is indeed a square.
Now, also note that segment $\overline{DG}$ goes from $D = (0,1)$ to $G = (2-e, 1)$, so $\overline{DG}$ is a **horizontal segment at** $y = 1$, from $x = 0$ to $x = 2-e$. So point $C = (1,1)$ lies on $\overline{DG}$ if and only if $0 \le 1 \le 2-e \Rightarrow e \le 1$, which is true since $0 < e < 1$. ✓ So this configuration satisfies the condition that C lies on segment $\overline{DG}$.
Step 5: Coordinates of all relevant points. Recall: $A = (0,0)$, $D = (0,1)$, $E = (1,e)$, $G = (2-e, 1)$. We need to find the **circumcenter** O of triangle AEG, and use the condition that A, D, and O are collinear.
Step 6: Find the circumcenter O of triangle AEG. Denote $O = (h,k)$. Since O is the circumcenter, it is equidistant from all three vertices:

$$h^2 + k^2 = (h-1)^2 + (k-e)^2 = (h-(2-e))^2 + (k-1)^2.$$

Equation 1: $OA^2 = OE^2$. Expanding the right side: $(h-1)^2 + (k-e)^2 = h^2 - 2h + 1 + k^2 - 2ek + e^2$. Setting it equal to $h^2 + k^2$ and canceling h^2, k^2 from both sides:

$$0 = -2h + 1 - 2ek + e^2 \implies 2h + 2ek = 1 + e^2. \quad \text{(Equation A)}$$

Equation 2: $OE^2 = OG^2$. The left side is $h^2 - 2h + 1 + k^2 - 2ek + e^2$; the right side is $h^2 - 2(2-e)h + (2-e)^2 + k^2 - 2k + 1$. Canceling h^2, k^2, 1 and bringing everything to the left:

$$-2h - 2ek + e^2 + 4h - 2eh + 2k - (4 - 4e + e^2) = 0,$$

which simplifies term by term ($-2h + 4h = 2h$; $-2ek - 2eh = -2e(k+h)$; $e^2 - e^2 = 0$) to

$$2h - 2e(h+k) + 2k - 4 + 4e = 0.$$

Dividing by 2 and grouping like terms: $(h+k)(1-e) + 2(e-1) = 0$. Since $e - 1 = -(1-e)$, this factors as

$$(1-e)\,[\,h + k - 2\,] = 0.$$

So either $1 - e = 0 \Rightarrow e = 1$, **but** this is **not allowed** since then $E = C$, which would collapse the square; or $h + k = 2$. (Equation B)
Step 7: Combine equations (A) and (B). From (B), $k = 2 - h$. Substituting into (A):

$$2h + 2e(2-h) = 1 + e^2 \implies 2h + 4e - 2eh = 1 + e^2 \implies 2h(1-e) = 1 + e^2 - 4e,$$

so

$$h = \frac{1 + e^2 - 4e}{2(1-e)}.$$

Step 8: Use collinearity of $A = (0,0)$, $D = (0,1)$, $O = (h,k)$. Points A and D are on the **vertical line** $x = 0$, so for O to be on line AD it must also satisfy $h = 0$. Setting the numerator to zero:

$$1 + e^2 - 4e = 0 \implies e^2 - 4e + 1 = 0 \implies e = \frac{4 \pm \sqrt{16-4}}{2} = 2 \pm \sqrt{3}.$$

Now $0 < e < 1$, so check values: $2 + \sqrt{3} \approx 3.732$ is too big; $2 - \sqrt{3} \approx 0.2679$ is valid. So $e = 2 - \sqrt{3}$.
Step 9: Find FG. Recall $F = (2-e,\ e)$ and $G = (2-e,\ 1)$. Segment FG is **vertical**, same x-coordinate, difference in y-coordinates:

$$FG = |1 - e| = 1 - e = 1 - (2 - \sqrt{3}) = \sqrt{3} - 1.$$

✓ **Final Answer:** $\boxed{\sqrt{3} - 1}$

Figure 8: Math case study from the smaller/larger setting, with the errors of the Qwen3-8B student marked in red.

Case Study: Code Generation (Smaller & Larger Setting)

Question: Given two lists `operator` and `operand`, where the first list contains basic algebra operations and the second is a list of integers, use the two lists to build an algebraic expression and return its evaluation, *e.g.,* `operator = ['+', '*', '-']` and `operand = [2, 3, 4, 5]` represent $2 + 3 \times 4 - 5 = 9$.

Qwen3-8B (student):
To solve this problem, we need to apply the given list of operators to the corresponding elements of the operand list in a sequential manner. Given that the length of the `operator` list is one less than the length of the `operand` list, we can construct the expression step-by-step and evaluate it.
Here's the step-by-step plan: 1. **Initialize the result** with the first operand.
2. **Iterate** over the operators and operands in pairs.
3. **Apply each operator** to the current result and the next operand.
4. **Return the final result** after all operations are completed.
Here is the Python code that implements the above logic:

```
def do_algebra(operator, operand):
    result = operand[0]
    for i in range(len(operator)):
        op = operator[i]
        num = operand[i + 1]
        if op == '+': result += num
        elif op == '-': result -= num
        elif op == '*': result *= num
        elif op == '//': result //= num
        elif op == '**': result **= num
    return result
```

Example Usage:

```
print(do_algebra(['+', '*', '-'], [2, 3, 4, 5])) # Output: 9
```

This code handles all the required operations and follows the constraints specified in the problem statement. (← evaluates strictly left to right, ignoring operator precedence: for `['+', '*', '-']` and `[2, 3, 4, 5]` the code computes $((2+3)\times 4)-5=15$, not the claimed 9 ($=2+3\times 4-5$), so even its own example comment is inconsistent with the code.)

W2S-OPD:
To solve this problem, we need to construct an algebraic expression using the given `operator` and `operand` lists, then evaluate it.
Plan: 1. The `operator` list has one fewer element than the `operand` list.
2. We can iterate through the `operator` list and apply each operator to consecutive elements from the `operand` list.
3. Use Python's `eval()` function to safely evaluate the constructed expression string.
4. Make sure the expression string is properly formatted with spaces to avoid any ambiguity or syntax errors.
Important Notes: We assume that the input is valid as per the problem constraints. We use `eval()` with caution, but in this context it's safe since the inputs are controlled and only contain allowed operations and integers. We format the expression string with spaces to ensure correct parsing.
Python Code:

```
def do_algebra(operator, operand):
    # Build the expression string
    expression = str(operand[0])
    for i in range(len(operator)):
        expression += f" {operator[i]} {operand[i + 1]}"
    # Evaluate the expression
    return eval(expression)
```

Example Usage:

```
operator = ['+', '*', '-']
operand = [2, 3, 4, 5]
print(do_algebra(operator, operand)) # Output: 9
```

Explanation: The expression string becomes `"2 + 3 * 4 - 5"`. Evaluating this gives $2+3\times 4-5=2+12-5=9$. This solution is concise, readable, and leverages Python's built-in capabilities to achieve the desired result.

Figure 9: Code case study under the smaller/larger setting, with the errors of the Qwen3-8B student marked in red.