Multi-Rollout On-Policy Distillation via Peer Successes and Failures

Anonymous Author(s)
Affiliation
Address
email

Abstract

Large language models are often post-trained with sparse verifier rewards, which indicate whether a sampled trajectory succeeds but provide limited guidance about where reasoning succeeds or fails. On-policy distillation (OPD) offers denser token-level supervision by training on student-generated trajectories, yet existing methods typically distill each rollout independently and ignore the other attempts sampled for the same prompt. We introduce Multi-Rollout On-Policy Distillation (MOPD), a peer-conditioned distillation framework that uses the student's local rollout group to construct more informative teacher signals. MOPD conditions the teacher on both successful and failed peer rollouts: successes provide positive evidence for valid reasoning patterns, while failures provide structured negative evidence about plausible mistakes to avoid. We study two peer-context constructions: positive peer imitation and contrastive success–failure conditioning. Experiments on competitive programming, mathematical reasoning, scientific question answering, and tool-use benchmarks show that MOPD consistently improves over standard on-policy baselines. Further teacher-signal analysis shows that mixed success–failure contexts better align teacher scores with verifier rewards, indicating that the gains arise from more faithful, instance-adaptive supervision. These results indicate that effective on-policy distillation should exploit the student's multi-rollout trial-and-error behavior rather than treating rollouts as isolated samples. Code is available at `https://anonymous.4open.science/r/mopd_code-C7BC/`.

1 Introduction

Large language models (LLMs) are often post-trained with reinforcement learning from verifiable rewards, where sampled solutions are scored by answer checkers, unit tests, or task-specific verifiers [18, 46, 48]. Although effective, these rewards are usually sparse: they indicate whether an entire trajectory succeeds or fails, but provide little guidance about which tokens or reasoning steps caused the outcome [69, 6]. Distillation and self-distillation provide a complementary form of supervision by converting teacher or privileged-model judgments into dense token-level signals [70, 17, 50, 7, 66]. On-policy distillation further aligns this supervision with the student's own behavior by training on trajectories sampled from the student policy [1, 14, 32].

Despite this advantage, existing OPD and on-policy self-distillation methods underuse a key source of information already available during training: multiple rollouts generated for the same problem instance. In many pipelines, each rollout is distilled independently, so the teacher signal for one trajectory is computed without access to the other attempts sampled from the same student. This design discards the local structure of the rollout group. For reasoning-intensive tasks, such local structure is highly informative: successful rollouts reveal valid solution strategies, while failed rollouts expose plausible but incorrect reasoning paths, missing constraints, formatting errors, or execution

Submitted to 40th Conference on Neural Information Processing Systems (NeurIPS 2026). Do not distribute.

mistakes. Treating rollouts independently prevents the teacher from comparing these alternatives and identifying where an unsuccessful trajectory diverges from a successful one.

We propose *Multi-Rollout On-Policy Distillation (MOPD)*, a peer-conditioned distillation framework that explicitly uses the student's rollout group to construct teacher signals. For each prompt, the student samples multiple on-policy trajectories, which are scored by a verifier and partitioned into successful and failed sets. MOPD then conditions the teacher on peer rollouts from the same problem instance when supervising a target trajectory. In this formulation, successful rollouts act as peer experts that provide positive evidence about valid reasoning patterns, while failed rollouts serve as structured negative evidence that identifies misleading solution paths the student should avoid.

The key insight is that **multi-rollout sampling creates a local trial-and-error space around each problem**. Instead of viewing each trajectory in isolation, the teacher can compare the current rollout against peer successes and failures, using this contrast to produce more targeted supervision. This turns OPD from independent imitation of individual trajectories into a comparative learning process: the teacher is no longer only a global expert, but also a local diagnostic model that can recognize instance-specific errors and distinguish superficially plausible failures from correct solutions. We instantiate this idea through two peer-context construction strategies: positive peer imitation and contrastive success–failure conditioning.

Moreover, prior methods often assume that a self-teacher becomes a reliable source of supervision once it is conditioned on privileged information, such as ground-truth outcomes, verified successful answers, hints, or environment feedback. However, this assumption is usually evaluated only indirectly through downstream training performance. Moreover, directly asking whether the privileged teacher can recover the correct answer is not a reliable diagnostic: if the privileged context already contains answer-related information or verifier feedback, the self-teacher may appear accurate by exploiting shortcuts rather than by producing a faithful supervision signal over the student's own trajectories. **What matters for self distillation is not merely whether the self-teacher knows the answer, but whether its token-level or trajectory-level preferences are aligned with correctness among the rollouts the student actually generates.**

To directly examine whether peer conditioning improves the self-teacher signal itself, we introduce an analysis of self-teacher signal quality. For each prompt, we fix a set of student-generated rollouts containing both successful and failed attempts, vary only the context shown to the self-teacher, and compare the self-teacher's normalized logits or scores with ground-truth verifier rewards. A better self-teacher signal should rank successful rollouts above failed ones, correlate more strongly with reward, and better separate correct from incorrect candidates. This analysis avoids the confound of answer-recovery tests and provides a direct measure of whether privileged contexts produce supervision that is actually aligned with correctness.

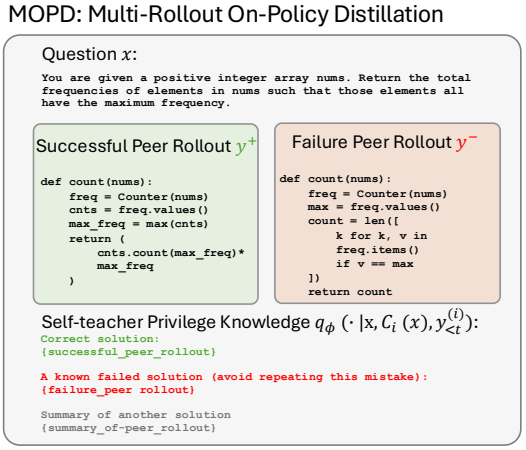

Figure 1: MOPD Illustration.

Empirically, MOPD improves over standard on-policy baselines across most scenarios on competitive programming, mathematical reasoning, scientific question answering, and tool-use benchmarks. The strongest gains arise from mixed peer contexts that combine successful and failed rollouts, supporting the hypothesis that contrastive evidence from peer rollout is more useful than positive demonstrations alone. Our self-teacher signal analysis further shows that the two-success-plus-one-failure context achieves the strongest alignment with verifier rewards across ranking and discrimination metrics, and our ablations show the same context yields the best compact downstream performance. These results indicate that effective distillation should exploit the student's multi-rollout trial-and-error behavior, not only the strength of the teacher.

2 Related Work

On-Policy Distillation Knowledge distillation transfers a teacher's predictive distribution to a smaller student, but off-policy training creates a distribution mismatch: the student is supervised on states it never visits at inference [20, 29, 44]. On-policy distillation addresses this by sampling trajectories from the student's own policy and computing teacher supervision on those same trajectories [1, 14, 32]. Within this framework, work has focused on divergence objectives, token reweighting, and interpolation between on- and off-policy regimes [56, 22, 72, 26]. A parallel thread replaces an external teacher with self-distillation: one copy of the model is conditioned on privileged context—solutions, hints, or environment feedback—and supervises a second copy that lacks it [23, 68, 78, 42, 65]. Information can also be aggregated across multiple sampled solutions through consistency, voting, or committee teachers [54, 40, 36]. However, the teacher signal for each rollout is constructed independently, leaving cross-rollout structure within a sampling group unexploited.

Multi-Rollout Reinforcement Learning and Verifier-Guided Improvement Post-training with verifiable rewards replaces learned reward models with programmatic checkers and updates the policy from group-relative advantages computed over multiple sampled rollouts per prompt [48, 55, 41, 43]. Process reward models provide denser credit assignment by grading intermediate reasoning steps [8, 47, 76], and hybrid pipelines couple these signals with on-policy distillation [60, 59, 75]. While these methods exploit multiple rollouts per prompt through advantage normalization or rejection sampling, the supervisory signal for any single trajectory is computed without conditioning on the contents of the others. The work most adjacent to ours treats successes and failures as complementary evidence at the level of advantages or filters, but stops short of routing both into a single peer-conditioned distillation target. Extended related work for both topics is in section F.

3 Preliminaries

We consider the problem of distilling reasoning capabilities from a teacher model into a student model under an on-policy sampling regime. Let $x \in \mathcal{X}$ denote an input problem or prompt, and let $y = (y_1, \ldots, y_T)$ denote an output trajectory generated autoregressively by a student policy π_θ. At decoding step t, we write the prefix as $y_{<t}$ and the token-level context as $c_t = (x, y_{<t})$. The student distribution is $\pi_\theta(\cdot \mid c_t)$, while the teacher distribution is denoted by $q_\phi(\cdot \mid c_t)$, possibly conditioned on additional privileged information. In prior self-distillation methods, the privileged information available to the teacher typically includes ground-truth outcomes, verified successful answers [67, 68, 78, 65], or environment feedback [23].

On-policy distillation. Offline distillation trains the student on trajectories sampled from a teacher or from a fixed dataset. This creates a distribution mismatch: during inference, the student may visit states that were not observed during training. On-policy distillation addresses this issue by first sampling trajectories from the student itself and then obtaining teacher supervision on those same trajectories. Formally, for each prompt x, the student generates $y \sim \pi_\theta(\cdot \mid x)$. The teacher then provides token-level targets along the student-induced trajectory. A standard on-policy distillation loss can be written as

$$\mathcal{L}_{\mathrm{OPD}} = \mathbb{E}_{x \sim \mathcal{D},\, y \sim \pi_\theta(\cdot \mid x)} \left[\sum_{t=1}^{T} D(\pi_\theta(\cdot \mid x, y_{<t}) \,\|\, q_\phi(\cdot \mid x, y_{<t})) \right], \tag{1}$$

where $D(\cdot)$ denotes a divergence loss, either KL, reverse KL, or Jensen-Shannon (JS), as detailed in section B. This formulation provides dense token-level supervision on the student's visited states.

Nevertheless, standard OPD treats each sampled trajectory independently. When multiple rollouts are generated for the same prompt, the teacher distribution for one rollout is usually computed without access to the successes and failures observed in the other rollouts. This prevents the teacher from exploiting local, instance-specific evidence contained in the rollout group.

4 Multi-Rollout On-Policy Distillation

We propose **Multi-Rollout On-Policy Distillation** (MOPD), a peer-conditioned distillation framework that exploits the local structure of multiple on-policy rollouts generated for the same question

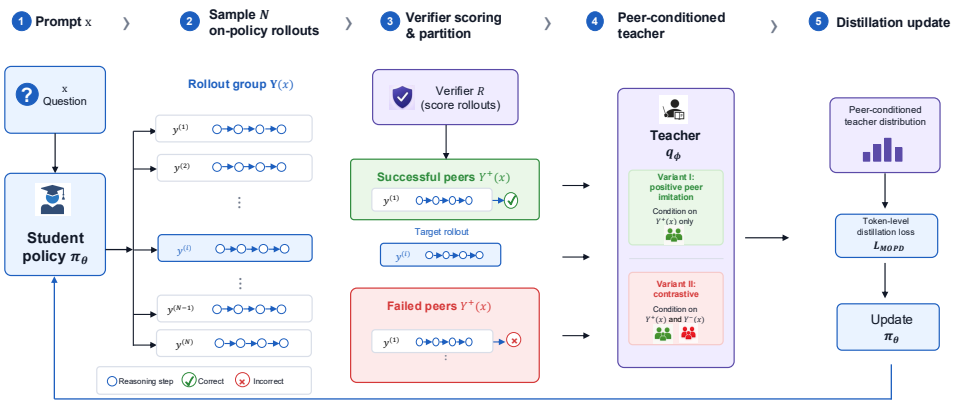

Figure 2: MOPD Pipeline.

as in fig. 2. Standard on-policy distillation supervises each sampled trajectory independently: the teacher evaluates the student's current trajectory without direct access to the alternative attempts produced for the same instance. This ignores a useful source of instance-specific information. In reasoning tasks, successful rollouts reveal valid solution paths, while failed rollouts expose plausible but incorrect branches. MOPD uses this local contrast to construct a more informative teacher signal.

At a high level, MOPD first samples a group of student rollouts for each prompt, scores them with a verifier, partitions them into successful and failed trajectories, and then conditions the teacher on peer information from the same rollout group. The resulting teacher distribution is no longer only a global distribution over next tokens; it becomes a *peer-conditioned* distribution that can compare the current trajectory against other local attempts. This allows the teacher to provide sharper token-level supervision, especially near branching points where failed trajectories diverge from successful ones.

4.1 Multi-rollout peer conditioning

For each prompt x, the student samples a group of N on-policy trajectories,

$$Y(x) = \{y^{(i)}\}_{i=1}^{N}, \qquad y^{(i)} \sim \pi_\theta(\cdot \mid x). \tag{2}$$

Each trajectory is evaluated by a verifier or reward function R, and the rollout group is partitioned according to a success threshold τ:

$$Y^+(x) = \{y^{(i)} : R(x, y^{(i)}) \geq \tau\}, \qquad Y^-(x) = \{y^{(i)} : R(x, y^{(i)}) < \tau\}. \tag{3}$$

Here $Y^+(x)$ contains peer successes and $Y^-(x)$ contains peer failures. For a target rollout $y^{(i)}$, we exclude the target itself and define

$$Y^+_{-i}(x) = Y^+(x) \setminus \{y^{(i)}\}, \qquad Y^-_{-i}(x) = Y^-(x) \setminus \{y^{(i)}\}. \tag{4}$$

MOPD constructs a peer context from these two sets:$C_i(x) = \mathcal{C}_v\left(x, y^{(i)}, Y^+_{-i}(x), Y^-_{-i}(x)\right)$, where $\mathcal{C}_v$ denotes a context construction rule. The index v specifies which peer-conditioning strategy is used. Given this context, the teacher distribution for the target trajectory becomes $q_\phi^{\text{peer}}(\cdot \mid x, y^{(i)}_{<t}) = q_\phi(\cdot \mid x, C_i(x), y^{(i)}_{<t})$. Compared with a standard teacher distribution $q_\phi(\cdot \mid x, y^{(i)}_{<t})$, the peer-conditioned teacher can use other rollouts from the same instance as local evidence. Successful peers provide positive support for correct reasoning patterns, while failed peers provide negative evidence about misleading or invalid trajectories.

The rollout-level MOPD loss is

$$\mathcal{L}^{(i)}_{\text{MOPD}} = \sum_{t=1}^{T_i} D\Big(\pi_\theta(\cdot \mid x, y^{(i)}_{<t}) \,\|\, q_\phi^{\text{peer}}(\cdot \mid x, y^{(i)}_{<t})\Big), \tag{5}$$

where D is a token-level divergence. In our implementation, D is instantiated as reverse KL for math tasks, and Jensen–Shannon divergence for other tasks.

4.2 Peer-context construction

The main design choice in MOPD is the construction of $C_i(x)$. We compare two natural choices: a positive-only context that exposes the teacher to additional successful peers, and a contrastive context that adds failed peers as structured negative evidence. This comparison isolates whether failure-as-negative-evidence sharpens the teacher signal beyond success-only conditioning.

Positive peer imitation. The first strategy conditions the teacher on successful peer trajectories only. Let $y^\star$ denote a primary successful rollout and let y^+ denote an additional successful peer, both selected from $Y_{-i}^+(x)$ when available. The context is

$$C_i^{(1)}(x) = \mathrm{Template}(y^\star) \oplus \mathrm{SuccessTemplate}(y^+), \tag{6}$$

where $\oplus$ denotes text concatenation. If only one successful peer is available, the context reduces to the primary successful trajectory:$C_i^{(1)}(x) = \mathrm{Template}(y^\star)$.

This variant treats successful rollouts as peer experts. It encourages the teacher to evaluate the target trajectory in light of alternative correct reasoning paths. Compared with using a single reference solution, positive peer imitation broadens the set of correct trajectories the teacher conditions on, so the distilled student is anchored to multiple valid derivation styles rather than one.

Contrastive success–failure conditioning. The second strategy conditions the teacher on both successful and failed peer trajectories. Given successful peers $y^\star, y^+ \in Y_{-i}^+(x)$ and a failed peer $y^- \in Y_{-i}^-(x)$, we define

$$C_i^{(2)}(x) = \mathrm{Template}(y^\star) \oplus \mathrm{SuccessTemplate}(y^+) \oplus \mathrm{FailureTemplate}(y^-). \tag{7}$$

This variant provides the teacher with contrastive local evidence. Successful rollouts indicate what the target trajectory should resemble, while failed rollouts identify plausible but incorrect reasoning patterns that should be avoided. The contrastive context can therefore sharpen token-level supervision by helping the teacher distinguish between superficially similar correct and incorrect trajectories. In tasks with sparse terminal rewards, this is especially useful because the failed peer exposes where an otherwise plausible solution path becomes invalid.

Unified view The two peer-context variants share a unified gated form: $C_i(x) = \mathrm{Template}(y^\star) \oplus \lambda_+ A_+(x) \oplus \lambda_- A_-(x)$, where $A_+(x)$ denotes additional successful demonstrations, $A_-(x)$ denotes failed demonstrations, and the gates $\lambda_+, \lambda_- \in \{0, 1\}$ determine which type of peer evidence is included. Algorithm 1 summarizes the MOPD training procedure.

Algorithm 1 MOPD Algorithm

Require: $\mathcal{D}$, π_θ, q_ϕ, R, N, τ, $\mathcal{C}_v$, α
Ensure: Updated student π_θ
for each training iteration **do**
  for each $x \in \mathcal{B}$ **do**
    Sample rollout group $Y(x)$ as in Eq. (2).
    Score and partition rollouts as in Eq. (3).
    Construct $C_i(x)$, query q_ϕ^{peer},
      compute $\mathcal{L}_{\mathrm{MOPD}}^{(i)}$.
  end for
  Update π_θ by minimizing
    $\mathcal{L}_{\mathrm{MOPD}} = \frac{1}{N}\sum_i \mathcal{L}_{\mathrm{MOPD}}^{(i)}$.
end for
return π_θ

5 Experiments

We evaluate whether augmenting the teacher with peer rollouts improves post-training in reasoning-intensive domains. Our method is compared against the base model, GRPO, and SDPO. SDPO conditions its self-teacher on environment feedback when available; our method instead conditions the teacher on peer rollouts—both successful and failed—from the same prompt. This allows us to test whether peer-augmented teacher information provides additional benefits beyond scalar-reward RL and feedback-conditioned self-distillation.

5.1 Experimental Setup

Benchmarks. We consider four families of tasks that require multi-step reasoning or precise execution. **Science QA.** We use the reasoning subsets (L3) from SciKnowEval [12], covering

undergraduate-level questions in biology, chemistry, physics, and materials science. **Tool Use.** We evaluate tool-use reasoning on ToolAlpaca [53], where the model must map a user request and an API specification to the correct tool call. **Code.** We evaluate on LiveCodeBench v6 (LCBv6) [24], which contains 131 contest-style programming problems released between February and May 2025. This split provides a recent and challenging evaluation of generalization to unseen coding problems. **Math.** For math training, we use a filtered subset of DeepMath [19], selecting 57K examples with difficulty level at least 6 following [65]. We evaluate on AIME2024, AIME2025, and HMMT25.

Metrics. For code and math, we report mean@8 and pass@8, following standard multi-sample evaluation. All results are reported as percentages. **Training protocol.** For each prompt, we sample a group of on-policy rollouts and evaluate them using a task-specific verifier or reward function. The resulting rollouts are partitioned into successful and failed sets. We then construct the privileged teacher context from peer rollouts, using two successful rollouts and one failed rollout unless otherwise specified. This design exposes the teacher to both correct solution patterns and informative failure modes. Across methods, we keep the rollout generation budget and optimization protocol fixed to ensure a fair comparison. **Implementation details.** Training is implemented in `verl` with asynchronous vLLM rollouts and FSDP-based optimization. We use a prompt batch size of 32 and sample 8 rollouts per prompt during training. The actor learning rate is set to 1×10^{-5} with 10 warmup steps. For generation, we use a maximum prompt length of 2048 tokens and a maximum response length of 4096 tokens (for math with 8192). Because SDPO conditions on additional solution context, we reserve an extra 2048-token headroom, yielding a total rollout context budget of 8192 tokens. Validation evaluates 8 samples per prompt. The configuration uses self-distillation with mixing coefficient $\alpha = 0.5$ and `distillation_topk`=100. We use FlashAttention-2 with bfloat16 model weights for both actor and critic. For training stability, we use reverse KL divergence for math tasks, and JS divergence otherwise.

5.2 Main Results for Self-Distillation

LiveCodeBench. Table 5 reports results on LCBv6 for Qwen3-4B and Qwen3-8B. LCBv6 provides structured environment feedback—such as compiler errors and test-case verdicts—at the end of each rollout. Our method consistently improves over the base model, GRPO, and SDPO at both model scales. On Qwen3-4B, our method improves over SDPO by large margin and on Qwen3-8B, our method reaches 61.82 mean@8 and 67.23 pass@8. **Math reasoning.** Table 1 shows results on AIME2024, AIME2025, and HMMT25 using Qwen3-4B as the base model. Our method substantially improves mean@8 compared to SDPO. SDPO is designed for settings with rich environment feedback, but math tasks provide only a single binary reward signal from a verifier, which likely explains why it underperforms the baseline in this domain. Peer-augmented supervision is particularly effective for mathematical reasoning in self-distillation, where alternative successful derivations and failed attempts provide complementary evidence about solution structure that a single verifier reward cannot. **Science QA.** Table 2 shows MOPD achieves the best performance across all four domains, on science multi-choice questions benchmarks. **Tool Use.** Tool Use requires precise mapping from natural-language requests to structured API calls, our method outperforms. Peer conditioning therefore extends beyond reasoning-rich settings to tasks where the primary challenge is accurate structured-output prediction. **Time Cost.** MOPD introduces acceptable computational overhead over SDPO, with total per-step time increasing by only 9.1% on LCB and 33.3% on QA-Chemistry, detailed breakdown results in table 9. **Number of ever-success questions during training.** As in fig. 3, MOPD reaches a higher count earlier and finally than SDPO, indicating faster exploration of the solution space across training prompts.

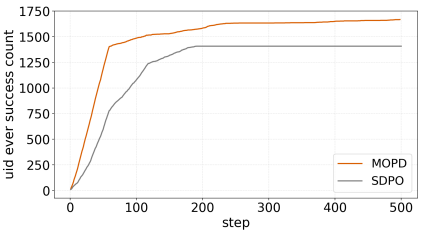

Figure 3: Number of training data that have ever generated a correct answer in the N rollout during training.

Case Study. During training, we save the generated rollouts and compare them on the same question across training steps to provide a case study. Additionally, after training for the same number of steps, we save checkpoints from both SDPO and MOPD, then sample from these checkpoints to evaluate whether each model can successfully solve the target problem and avoid failure patterns. The details in section C, showing that **MOPD and SDPO start from the same failure pattern early in training,**

Table 1: Math reasoning results on AIME2024/2025 and HMMT25. All methods start from the same Qwen3-4B model and use the same generation budget. Bold marks the best result in each column.

	AIME2025		AIME2024		HMMT25 Feb.		HMMT25 Nov.	
Method	mean@8	pass@8	mean@8	pass@8	mean@8	pass@8	mean@8	pass@8
Qwen3-4B	17.92	32.57	16.89	31.60	5.06	10.00	9.58	13.91
GRPO	**25.80**	**36.66**	17.09	32.05	16.92	**20.94**	13.16	19.53
SDPO	7.81	16.66	7.29	16.29	8.59	13.61	12.01	18.49
MOPD	25.41	36.28	**28.54**	**33.12**	**18.50**	19.58	**15.83**	**22.47**

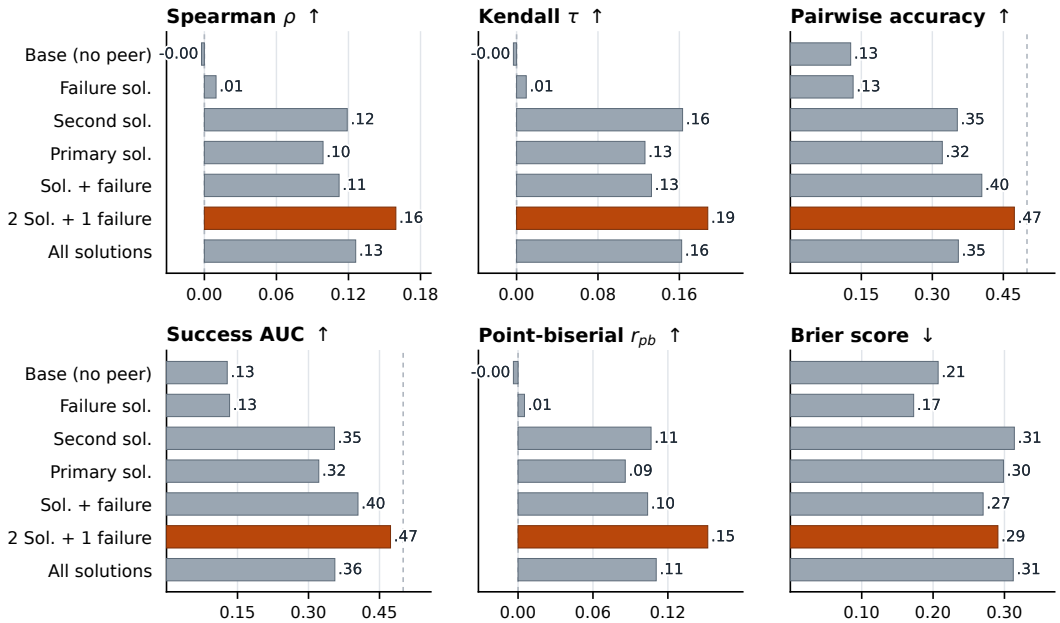

Figure 4: Self-teacher-signal quality across seven context conditions. Each panel reports an averaged prompt-level metric. Higher is better for all metrics except the Brier score.

but by repeatedly exposing the teacher to failed peers of the same bug family and explicitly signaling 'avoid this mistake,' MOPD suppresses the error by checkpoint time — while baseline method, lacking this channel, does not.

5.3 Analysis of Self-Teacher Signal Quality

Setup: self-teacher signal evaluation. To understand why peer information improves training, we directly analyze the quality of the self-teacher signal induced by different context constructions. For each question x, we sample a set of rollouts $\{y^{(i)}\}_{i=1}^{n}$ from the policy of the base model and evaluate each response using both the task verifier and the self-teacher model. Each response is associated with three quantities: $(y^{(i)}, s_i, r_i)$, where $s_i \in \mathbb{R}$ is the self-teacher average token logprob $s_i = \frac{1}{T_i}\sum_{t=1}^{T_i} \log q_\phi\left(y_t^{(i)} \mid x,\, C_i(x),\, y_{<t}^{(i)}\right)$ assigned to response $(y^{(i)}$, $r_i \in [0,1]$ is the ground truth reward. The self-teacher score s_i is computed under each specific teacher privilege knowledge, while r_i is a fixed property of the candidate rollout. Thus, by holding the prompt and candidate responses fixed and varying only the teacher-visible context, we can isolate how different types of peer information affect the reliability of the self-teacher signal. We consider several context conditions, including base (no peer information), successful peer rollouts, failed peer rollouts, combinations of successful and failed rollouts, and all the rollouts. The analysis uses Qwen3-4B model generated on the LiveCodeBench-v6 benchmark.

We evaluate each self-teacher context condition with six complementary metrics covering ranking, discrimination, and calibration; full definitions are in section A. **Mean Spearman Correlation** and **Mean Kendall's** τ both measure ordinal agreement between teacher scores and ground-truth rewards

Table 2: Science QA results.

Method	Bio.	Chem.	Phys.	Mat.
Qwen3-8B	30.89	42.98	58.44	65.59
GRPO	47.32	64.24	64.12	72.09
SDPO	50.60	62.91	67.36	72.34
MOPD	**55.69**	**74.29**	**76.25**	**78.59**

Table 3: ToolUse results.

Method	Qwen3-8B	GRPO	SDPO	MOPD
Tool Use	59.11	60.32	63.45	**66.73**

Table 4: Ablation on peer-context construction for MOPD: which combination of successful and failed peer rollouts to inject into the self-teacher context.

	QA-Chemistry		LiveCodeBench	
Peer context $\mathcal{C}(u)$	mean@8	pass@8	mean@8	pass@8
1 successful solution	51.90	58.92	49.49	64.34
2 successful solutions	52.62	59.02	55.62	64.02
1 failure solution	35.67	37.64	46.72	54.86
1 success + 1 failure	60.50	68.12	58.90	62.01
2 success + 1 failure	74.29	83.53	61.82	67.23
8 solutions (2 hours)	73.57	83.11	58.43	63.68
8 solutions (4 hours)	81.25	84.49	58.95	64.71

Table 5: LCBv6 results.

	Qwen3-4B		Qwen3-8B	
Method	**mean@8**	**pass@8**	**mean@8**	**pass@8**
Base	28.80	49.36	30.97	53.02
GRPO	40.75	55.43	43.65	58.72
SDPO	48.84	63.23	49.49	64.34
Ours	**53.67**	**65.48**	**61.82**	**67.23**

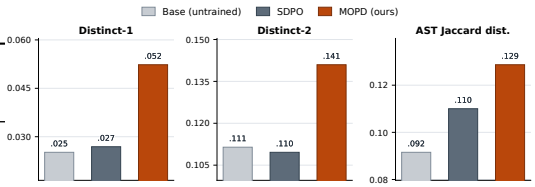

Figure 5: Diversity Analysis.

across rollouts. **Pairwise Accuracy** directly reports the fraction of reward-distinguishable rollout pairs that the teacher orders correctly. **Success AUC** measures the probability that a successful rollout receives a higher teacher score than a failed one; **Success Point-Biserial Correlation** quantifies the linear association between teacher scores and binary success labels; and **Success Brier Score (Sigmoid)** assesses probabilistic calibration by mapping teacher scores through a sigmoid and computing MSE against rewards.

Results and interpretation. Figure 4 and Table 4 together reveal how peer context composition affects teacher signal quality and downstream performance. 1) Successful peer rollouts are the primary driver of teacher-signal quality: any condition containing at least one successful peer lifts pairwise accuracy from 0.13 at the no-peer baseline to above 0.32. A failed peer alone barely moves the discrimination metrics. 2) Successful peers have a stronger effect on correctness-related metrics—Spearman ρ rises from 0.00 to above 0.10 and Kendall τ from 0.00 to above 0.13 when at least one successful solution is included—suggesting that seeing a correct peer helps the teacher better assess the target rollout's quality. 3) Failed peers complement successful ones: adding a failure to a success-only context lifts pairwise accuracy from 0.32 to 0.40, showing that negative evidence sharpens decision boundaries that positive evidence alone leaves blurred. 4) Combining both types yields the best results: the "2 success + 1 failure" context achieves the highest score on 5 of the 6 ranking and discrimination metrics in the signal-quality analysis, with a competitive Brier score, and the highest LCB downstream mean@8 among the compact peer-context settings. 5) We notice that the results for base are very low, for example, the pairwise accuracy and success AUC is only 0.13. We look into the generations of Qwen3-4B and find that **57%** of prompts have no successful rollout ($Y^+(x) = \emptyset$), explaining the near-zero metrics in the hard bucket. This confirms that correct solutions occupy extremely low-probability regions of the model's output distribution—the self-teacher finds them genuinely surprising, which directly explains why model accuracy remains below 30% on lcb-v6. 6) On LiveCodeBench, providing all available rollouts yields no additional gains beyond the compact two-success-plus-one-failure context; more peer rollouts do not strictly dominate a balanced contrastive pair on every benchmark.

5.4 Solution Diversity Analysis

We investigate whether MOPD's exposure to multiple peer rollouts during training leads to greater diversity in generated solutions. We measure two axes of diversity that are well defined for code: Distinct-1 and Distinct-2 for within-response lexical richness (the vocabulary and bigram coverage

inside a single program), and AST-node Jaccard distance for between-response structural variation (the difference between programs in their parsed syntax-tree composition). Both are computed on LiveCodeBench-v6 checkpoints at step 50, with each model generating 8 responses per problem. Detailed implementation and metric definitions are provided in section D.

Figure 5 shows that MOPD produces substantially more diverse outputs than both the base model and SDPO across all three metrics. On Distinct-1 and Distinct-2, SDPO remains close to the untrained base model (0.027 vs. 0.025; 0.110 vs. 0.111), suggesting that its training signal does not encourage lexical variation. MOPD, by contrast, achieves nearly double the Distinct-1 score of SDPO (0.052 vs. 0.027) and a notable gain in Distinct-2 (0.141 vs. 0.110). A similar pattern holds at the structural level: AST Jaccard distance increases from 0.110 (SDPO) to 0.129 (MOPD), indicating that MOPD generates solutions with more structurally distinct implementations. Conditioning the teacher on failed peer rollouts therefore improves both correctness and the lexical and structural diversity of the student's outputs.

5.5 Does Peer Information Improve Teacher–Student Distillation?

We next study whether peer information remains useful in a teacher–student setting, where the supervision signal is provided by a larger teacher model and used to train a smaller student. Our general knowledge distillation (GKD) pipeline follows an offline teacher-forcing paradigm: for each input prompt, a larger teacher model generates 8 completions, which are then converted into supervised fine-tuning examples. The student is trained by maximum-likelihood learning on these teacher responses. In on-policy distillation (OPD), the teacher provides supervision conditioned on on-policy student responses. Ours TS method further augments this teacher context with peer information: successful and failed peer rollouts.

Table 6: Teacher–student on-policy distillation results. All methods share the same rollout budget and hyperparameter.

	ToolUse		**LiveCodeBench**	
Method	mean@8	pass@8	mean@8	pass@8
Qwen3-4B (S)	53.27	58.58	28.80	49.36
Qwen3-14B (T)	56.02	60.89	46.75	65.16
GKD (TS)	55.91	60.05	46.05	56.67
OPD (TS)	59.63	61.58	60.92	67.30
SDPO (Self)	62.04	65.97	48.84	63.23
Ours (TS)	**66.44**	**68.61**	**61.92**	**67.35**
Ours (Self)	64.61	66.81	53.67	65.48

Table 6 reveals several trends in peer-conditioned distillation. 1) On ToolUse and LiveCodeBench, traditional teacher-student setups with on-policy strategies still outperform self-distillation. 2) When the teacher is given richer context through peer information, distillation can surpass the teacher model's own performance—a result that underscores the value of privileged observations at inference time. 3) Adding peer information consistently improves both the self-distillation and teacher–student settings, confirming that the benefit is not specific to a particular pairing of model sizes. 4) On-policy methods outperform their off-policy counterparts across both domains, consistent with the distribution-shift argument motivating on-policy training.

6 Conclusion

We introduced MOPD, a peer-conditioned on-policy distillation framework that uses multiple student rollouts from the same prompt to construct more informative teacher signals. By conditioning the teacher on successful and failed peer rollouts, MOPD turns independent trajectory supervision into a comparative learning process: successes provide positive evidence for valid reasoning, while failures expose error patterns to avoid. Experiments across code, math, science QA, and tool use show that MOPD consistently improves over standard on-policy baselines. Both ablations and teacher-signal analysis indicate that mixed success–failure contexts produce the most faithful supervision, better aligning teacher scores with verifier rewards and downstream performance. Effective distillation should exploit the student's rollout group as a structured collection of successes and failures, rather than treating each trajectory in isolation.

References

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

[2] Rong Bao, Donglei Yu, Kai Fan, and Minpeng Liao. Fixing distribution shifts of llm self-critique via on-policy self-play training. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 17680–17700, 2025.

[3] Nicolas Boizard, Kevin El Haddad, Céline Hudelot, and Pierre Colombo. Towards Cross-Tokenizer Distillation: the Universal Logit Distillation Loss for LLMs. *Transactions on Machine Learning Research*, 2025.

[4] Dan Busbridge, Amitis Shidani, Floris Weers, Jason Ramapuram, Etai Littwin, and Russ Webb. Distillation Scaling Laws. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2025.

[5] Stephen Casper, Xander Davies, Claudia Shi, Thomas Krendl Gilbert, Jérémy Scheurer, Javier Rando, Rachel Freedman, Tomasz Korbak, David Lindner, Pedro Freire, et al. Open problems and fundamental limitations of reinforcement learning from human feedback. *Transactions on Machine Learning Research*, 2023. Survey Certification, Featured Certification.

[6] Alex J Chan, Hao Sun, Samuel Holt, and Mihaela Van Der Schaar. Dense reward for free in reinforcement learning from human feedback. *arXiv preprint arXiv:2402.00782*, 2024.

[7] Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. Self-play fine-tuning converts weak language models to strong language models. In *International Conference on Machine Learning*, pages 6621–6642. PMLR, 2024.

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

[9] Qiyuan Deng, Kehai Chen, Min Zhang, and Zhongwen Xu. HiPO: Self-hint policy optimization for RLVR. In *The Fourteenth International Conference on Learning Representations*, 2026.

[10] Ken Ding. HDPO: Hybrid Distillation Policy Optimization via Privileged Self-Distillation. *arXiv preprint arXiv:2603.23871*, 2026.

[11] Aradhya Dixit, Tianxi Liang, and Jai Telang. Project aletheia: Verifier-guided distillation of backtracking for small language models. In *Logical and Symbolic Reasoning in Language Models @ AAAI 2026*, 2026.

[12] Yike Feng and et al. Sciknoweval: Evaluating multi-level scientific knowledge of large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 2024.

[13] Yuqian Fu, Tinghong Chen, Jiajun Chai, Xihuai Wang, Songjun Tu, Guojun Yin, Wei Lin, Qichao Zhang, Yuanheng Zhu, and Dongbin Zhao. SRFT: A single-stage method with supervised and reinforcement fine-tuning for reasoning. In *The Fourteenth International Conference on Learning Representations*, 2026.

[14] Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. MiniLLM: On-Policy Distillation of Large Language Models. *Proceedings of ICLR*, 2024.

[15] Xinyan Guan, Yanjiang Liu, Xinyu Lu, Boxi Cao, Ben He, Xianpei Han, Le Sun, Jie Lou, Bowen Yu, Yaojie Lu, et al. Search, verify and feedback: Towards next generation post-training paradigm of foundation models via verifier engineering. *arXiv preprint arXiv:2411.11504*, 2024.

[16] Arnav Gudibande, Eric Wallace, Charlie Snell, Xinyang Geng, Hao Liu, Pieter Abbeel, Sergey Levine, and Dawn Song. The False Promise of Imitating Proprietary LLMs. *arXiv preprint arXiv:2305.15717*, 2023.

[17] Caglar Gulcehre, Ziyu Wang, Alexander Novikov, Sarah Bechtle, Nicolas Heess, Sergey Pankov, Nan Rosemary Ke, Jean-Baptiste Alayrac, Victor Bapst, Rishabh Hoffmann, et al. Rest: Reinforced self-training (rest) for language models. *arXiv preprint arXiv:2308.08998*, 2023.

[18] 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.

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

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

[21] Jiwoo Hong, Noah Lee, and James Thorne. Orpo: Monolithic preference optimization without reference model. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 11170–11189, 2024.

[22] Haiduo Huang, Jiangcheng Song, Yadong Zhang, and Pengju Ren. SelecTKD: Selective Token-Weighted Knowledge Distillation for LLMs. *arXiv preprint arXiv:2510.24021*, 2025.

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

[24] 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. In *The Thirteenth International Conference on Learning Representations*, 2025.

[25] Yuxin Jiang, Chunkit Chan, Mingyang Chen, and Wei Wang. Lion: Adversarial Distillation of Proprietary Large Language Models. *Proceedings of EMNLP*, 2023.

[26] 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.

[27] Seongryong Jung, Suwan Yoon, DongGeon Kim, and Hwanhee Lee. ToDi: Token-wise Distillation via Fine-Grained Divergence Control. *Proceedings of EMNLP*, 2025.

[28] 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.

[29] Yoon Kim and Alexander M. Rush. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1317–1327, 2016.

[30] 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.

[31] Jongwoo Ko, Tianyi Chen, Sungnyun Kim, Tianyu Ding, Luming Liang, Ilya Zharkov, and Se-Young Yun. DistiLLM-2: A Contrastive Approach Boosts the Distillation of LLMs. *Proceedings of ICML*, 2025.

[32] Jongwoo Ko, Sungnyun Kim, Tianyi Chen, and Se-Young Yun. DistiLLM: Towards Streamlined Distillation for Large Language Models. *Proceedings of ICML*, 2024.

[33] Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and William B Dolan. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 110–119, 2016.

[34] Yaxuan Li, Yuxin Zuo, Bingxiang He, Jinqian Zhang, Chaojun Xiao, Cheng Qian, Tianyu Yu, Huan ang Gao, Wenkai Yang, Zhiyuan Liu, and Ning Ding. Rethinking on-policy distillation of large language models: Phenomenology, mechanism, and recipe, 2026.

[35] Yingru Li, Ziniu Li, and Jiacai Liu. A Note on Hybrid Online Reinforcement and Imitation Learning for LLMs: Formulations and Algorithms. *arXiv preprint arXiv:2512.23097*, 2025.

[36] Zhuochun Li, Yuelyu Ji, Rui Meng, and Daqing He. Learning from committee: Reasoning distillation from a mixture of teachers with peer-review. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 4190–4205, 2025.

[37] Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's Verify Step by Step. *Proceedings of ICLR*, 2024.

[38] Yuanjie Lyu, Chengyu Wang, Jun Huang, and Tong Xu. From Correction to Mastery: Reinforced Distillation of Large Language Model Agents. *arXiv preprint arXiv:2509.14257*, 2025.

[39] Yu Meng, Mengzhou Xia, and Danqi Chen. Simpo: Simple preference optimization with a reference-free reward. *Advances in Neural Information Processing Systems*, 37:124198–124235, 2024.

[40] Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*, 2025.

[41] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

[42] Emiliano Penaloza, Dheeraj Vattikonda, Nicolas Gontier, Alexandre Lacoste, Laurent Charlin, and Massimo Caccia. Privileged Information Distillation for Language Models. *arXiv preprint arXiv:2602.04942*, 2026.

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

[44] Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, pages 627–635, 2011.

[45] Hejian Sang, Yuanda Xu, Zhengze Zhou, Ran He, Zhipeng Wang, and Jiachen Sun. On-Policy Self-Distillation for Reasoning Compression. *arXiv preprint arXiv:2603.05433*, 2026.

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

[47] Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. Rewarding progress: Scaling automated process verifiers for LLM reasoning. In *The Thirteenth International Conference on Learning Representations*, 2025.

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

[49] Idan Shenfeld, Mehul Damani, Jonas Hübotter, and Pulkit Agrawal. Self-Distillation Enables Continual Learning. *arXiv preprint arXiv:2601.19897*, 2026.

[50] Avi Singh, John D Co-Reyes, Rishabh Agarwal, Ankesh Anand, Piyush Patil, Xavier Garcia, Peter J Liu, James Harrison, Jaehoon Lee, Kelvin Xu, et al. Beyond human data: Scaling self-training for problem-solving with language models. *Transactions on Machine Learning Research*, 2024. Expert Certification.

[51] Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

[52] Mingyang Song and Mao Zheng. A survey of on-policy distillation for large language models, 2026.

[53] Qiaoyu Tang and et al. Toolalpaca: Generalized tool learning for language models with 3000 simulated cases. *arXiv preprint arXiv:2306.05301*, 2023.

[54] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023.

[55] Xumeng Wen, Zihan Liu, Shun Zheng, Shengyu Ye, Zhirong Wu, Yang Wang, Zhijian Xu, Xiao Liang, Junjie Li, Ziming Miao, Jiang Bian, and Mao Yang. Reinforcement learning with verifiable rewards implicitly incentivizes correct reasoning in base LLMs. In *The Fourteenth International Conference on Learning Representations*, 2026.

[56] Yuqiao Wen, Zichao Li, Wenyu Du, and Lili Mou. f-Divergence Minimization for Sequence-Level Knowledge Distillation. *Proceedings of ACL*, 2023.

[57] Fang Wu, Weihao Xuan, Ximing Lu, Mingjie Liu, Yi Dong, Zaid Harchaoui, and Yejin Choi. The invisible leash: Why rlvr may or may not escape its origin. *arXiv preprint arXiv:2507.14843*, 2025.

[58] Taiqiang Wu, Chaofan Tao, Jiahao Wang, Runming Yang, Zhe Zhao, and Ngai Wong. Rethinking Kullback-Leibler Divergence in Knowledge Distillation for Large Language Models. *Proceedings of COLING*, 2025.

[59] Hongling Xu, Qi Zhu, Heyuan Deng, Jinpeng Li, Lu Hou, Yasheng Wang, et al. KDRL: Post-Training Reasoning LLMs via Unified Knowledge Distillation and Reinforcement Learning. *arXiv preprint arXiv:2506.02208*, 2025.

[60] Shicheng Xu, Liang Pang, Yunchang Zhu, Jia Gu, Zihao Wei, Jingcheng Deng, Feiyang Pan, Huawei Shen, and Xueqi Cheng. RLKD: Distilling LLMs' Reasoning via Reinforcement Learning. *arXiv preprint arXiv:2505.16142*, 2025.

[61] Wenda Xu, Rujun Han, Zifeng Wang, Long T. Le, Dhruv Madeka, Lei Li, William Yang Wang, Rishabh Agarwal, Chen-Yu Lee, and Tomas Pfister. Speculative Knowledge Distillation: Bridging the Teacher-Student Gap Through Interleaved Sampling. *Proceedings of ICLR*, 2025.

[62] Xiaohan Xu, Ming Li, Chongyang Tao, Tao Shen, Reynold Cheng, Jinyang Li, Can Xu, Dacheng Tao, and Tianyi Zhou. A Survey on Knowledge Distillation of Large Language Models. *arXiv preprint arXiv:2402.13116*, 2024.

[63] Jianhao Yan, Yafu Li, Zican Hu, Zhi Wang, Ganqu Cui, Xiaoye Qu, Yu Cheng, and Yue Zhang. Learning to Reason under Off-Policy Guidance. *arXiv preprint arXiv:2504.14945*, 2025.

[64] 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, et al. Qwen3 Technical Report. *arXiv preprint arXiv:2505.09388*, 2025.

[65] 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.

[66] Zhaorui Yang, Tianyu Pang, Haozhe Feng, Han Wang, Wei Chen, Minfeng Zhu, and Qian Liu. Self-distillation bridges distribution gap in language model fine-tuning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1028–1043, 2024.

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

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

[69] Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in LLMs beyond the base model? In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[70] Eric Zelikman, Yuhuai Wu, Jesse Laskin, and Noah D Goodman. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488, 2022.

[71] Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. Rest-mcts*: Llm self-training via process reward guided tree search. *Advances in Neural Information Processing Systems*, 37:64735–64772, 2024.

[72] Dongxu Zhang, Zhichao Yang, Sepehr Janghorbani, Jun Han, Andrew Ressler, Qian Qian, Gregory D. Lyng, Sanjit Singh Batra, and Robert E. Tillman. Fast and Effective On-policy Distillation from Reasoning Prefixes. *arXiv preprint arXiv:2602.15260*, 2026.

[73] Songming Zhang, Xue Zhang, Tong Zhang, Bojie Hu, Yufeng Chen, and Jinan Xu. AlignDistil: Token-Level Language Model Alignment as Adaptive Policy Distillation. *Proceedings of ACL*, 2025.

[74] Xiaoying Zhang, Yipeng Zhang, Hao Sun, Kaituo Feng, Chaochao Lu, Chao Yang, and Helen Meng. Critique-grpo: Advancing llm reasoning with natural language and numerical feedback. *arXiv preprint arXiv:2506.03106*, 2025.

[75] Zhaoyang Zhang, Shuli Jiang, Yantao Shen, Yuting Zhang, Dhananjay Ram, Shuo Yang, et al. Reinforcement-aware Knowledge Distillation for LLM Reasoning. *arXiv preprint arXiv:2602.22495*, 2026.

[76] Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. The lessons of developing process reward models in mathematical reasoning. *arXiv preprint arXiv:2501.07301*, 2025.

[77] Shiwan Zhao, Zhihu Wang, Xuyang Zhao, Jiaming Zhou, Caiyue Xu, Chenfei Liu, Liting Zhang, Yuhang Jia, Yanzhe Zhang, Hualong Yu, Zichen Xu, Qicheng Li, and Yong Qin. Large language model post-training: A unified view of off-policy and on-policy learning, 2026.

[78] 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.

A Teacher Signal Quality Metric

Mean Spearman Correlation. Spearman correlation measures the agreement between the rank ordering induced by teacher scores and the rank ordering induced by rewards. For a prompt, let $\mathrm{rank}(s_i)$ and $\mathrm{rank}(r_i)$ denote the average ranks under ties. The prompt-level Spearman correlation is

$$\rho_{\mathrm{S}} \;=\; \mathrm{corr}\big(\mathrm{rank}(s_1), \ldots, \mathrm{rank}(s_n),\ \mathrm{rank}(r_1), \ldots, \mathrm{rank}(r_n)\big),$$

where $\mathrm{corr}(\cdot, \cdot)$ is the Pearson correlation coefficient. We report the mean of this quantity across prompts. Higher values indicate that the teacher better preserves the reward-induced ordering.

Mean Kendall's τ. Kendall's τ evaluates pairwise ranking consistency between teacher scores and rewards. For each unordered pair (i, j) with $i < j$, let the pair be *concordant* if $(s_i - s_j)(r_i - r_j) > 0$ and *discordant* if $(s_i - s_j)(r_i - r_j) < 0$. Pairs tied in either score or reward are ignored. The prompt-level metric is

$$\tau \;=\; \frac{C - D}{C + D},$$

where C and D are the numbers of concordant and discordant pairs, respectively. We then average τ across prompts. Higher values indicate stronger ordinal agreement.

Pairwise Accuracy. Pairwise accuracy measures the fraction of unequal-reward pairs that are ordered correctly by the teacher. Formally,

$$\mathrm{PairAcc} \;=\; \frac{\sum_{i<j} \mathbf{1}[r_i \neq r_j]\mathbf{1}[(s_i - s_j)(r_i - r_j) > 0]}{\sum_{i<j} \mathbf{1}[r_i \neq r_j]}.$$

Unlike Kendall's τ, this metric does not penalize incorrect pairs symmetrically around zero; instead, it directly reports the proportion of reward-distinguishable pairs ranked correctly. Higher is better.

Success AUC. Success AUC treats the teacher score as a binary classifier score for success. Let $\mathcal{P} = \{i : y_i = 1\}$ and $\mathcal{N} = \{j : y_j = 0\}$. The prompt-level AUC is computed by comparing all positive–negative pairs:

$$\mathrm{AUC} \;=\; \frac{1}{|\mathcal{P}||\mathcal{N}|} \sum_{i \in \mathcal{P}} \sum_{j \in \mathcal{N}} \left(\mathbf{1}[s_i > s_j] + \frac{1}{2}\mathbf{1}[s_i = s_j] \right).$$

This is the probability that a randomly chosen successful response receives a higher teacher score than a randomly chosen unsuccessful one, with ties receiving half credit. Higher is better.

Success Point-Biserial Correlation. To quantify the linear association between teacher scores and binary success labels, we compute the point-biserial correlation, which is simply the Pearson correlation between the continuous teacher scores and the binary labels:

$$\rho_{\mathrm{PB}} \;=\; \mathrm{corr}(s_1, \ldots, s_n,\ r_1, \ldots, r_n).$$

A larger positive value indicates that successful responses tend to receive higher teacher scores.

Success Brier Score (Sigmoid). To assess probabilistic calibration of teacher scores with respect to success, we first map each score to a pseudo-probability using the sigmoid transform

$$p_i \;=\; \sigma(s_i) \;=\; \frac{1}{1 + e^{-s_i}},$$

and then compute the Brier score

$$\mathrm{Brier} \;=\; \frac{1}{n} \sum_{i=1}^{n} (p_i - r_i)^2.$$

This metric measures the mean squared error between predicted success probabilities and observed binary outcomes. Lower values indicate better calibration.

Condition	Mean Spearman	Mean Kendall's τ	Pairwise Acc.	Success AUC	Success Point-Biserial	Success Brier (Sig.)
Base	-0.0024	-0.0031	0.1275	0.1283	-0.0037	0.2071
Primary Sol.	0.0988	0.1262	0.3211	0.3217	0.0859	0.2986
Second Sol.	0.1190	0.1632	0.3528	0.3547	0.1065	0.3139
Failure Sol.	0.0097	0.0095	0.1326	0.1332	0.0052	0.1728
Sol.+Failure	0.1122	0.1326	0.4045	0.4045	0.1038	0.2701
2 Suc.+1 Failure	0.1595	0.1879	0.4733	0.4733	0.1520	0.2908
All Solutions	0.1260	0.1623	0.3550	0.3556	0.1107	0.3124

Table 7: Prompt-level offline teacher-signal metrics averaged over prompts for the six context conditions. Higher is better for all metrics except Success Brier (Sigmoid), where lower is better.

Table 7 reports prompt-level teacher-signal metrics across the six context conditions. The *2 Suc.+1 Failure* condition consistently achieves the strongest performance, attaining the highest Mean Spearman (0.1595), Mean Kendall's τ (0.1879), Pairwise Accuracy (0.4733), Success AUC (0.4733), and Success Point-Biserial (0.1520), indicating that a mixed context of two successful and one failed solution provides the richest teacher signal. In contrast, the *Failure Sol.* condition performs the worst across nearly all ranking and discrimination metrics, with a Mean Spearman of only 0.0097 and a Pairwise Accuracy of 0.1326, suggesting that failure demonstrations alone offer little useful signal. *Failure Sol.* attains the lowest Brier score (0.1728), but its discrimination metrics also collapse to near-zero, so this calibration value reflects a degenerate near-uniform output rather than a useful signal. Among the success-only conditions, *Second Sol.* slightly outperforms *Primary Sol.* and *All Solutions* across most metrics, while *Sol.+Failure* improves pairwise accuracy over either success-only condition alone.

B Preliminary on Divergence-based distillation

Divergence-based distillation. A common approach to distillation is to minimize a divergence between the teacher distribution q_ϕ and the student distribution π_θ at each token position. In language model distillation, the direction of the KL divergence induces qualitatively different behavior.

The *forward KL* objective,

$$D_{\mathrm{KL}}(q_\phi(\cdot \mid c_t) \,\|\, \pi_\theta(\cdot \mid c_t)), \tag{8}$$

penalizes the student for assigning low probability to tokens favored by the teacher. It is therefore mode-covering: when the teacher assigns mass to multiple plausible continuations, forward KL encourages the student to preserve this diversity.

The *reverse KL* objective,

$$D_{\mathrm{KL}}(\pi_\theta(\cdot \mid c_t) \,\|\, q_\phi(\cdot \mid c_t)), \tag{9}$$

penalizes the student for placing probability mass on tokens that the teacher considers unlikely. This objective is mode-seeking and is often attractive in on-policy settings because the expectation is taken over the student distribution. However, reverse KL may over-concentrate probability mass on a subset of teacher-preferred modes, especially when the teacher distribution is uncertain or when the supervision signal is incomplete.

A symmetric alternative is the Jensen–Shannon divergence,

$$D_{\mathrm{JS}}(p \,\|\, q) = \frac{1}{2} D_{\mathrm{KL}}(p \,\|\, m) + \frac{1}{2} D_{\mathrm{KL}}(q \,\|\, m), \qquad m = \frac{1}{2}(p + q). \tag{10}$$

D_{JS} is symmetric and bounded. In principle, it can provide a more stable compromise between mode-covering and mode-seeking behavior. In practice, however, computing such distribution-level objectives over the full vocabulary may be expensive for large language models, motivating approximations based on sampled tokens, top-k support, or teacher-provided log-probabilities.

C Case Studies

We present three complementary forms of evidence for MOPD's qualitative advantage. Case Studies 1 and 2 isolate checkpoint-level semantic failures: both models see the same task, but MOPD

preserves the crucial constraints that SDPO drops. Case Study 1 additionally provides direct training-time evidence linking the checkpoint gap to MOPD's peer-conditioning mechanism. Case Study 3 examines training dynamics, showing that MOPD internalizes correct behavior earlier while SDPO remains unstable late in training.

C.1 Case Study 1: Identifier-space composition (checkpoint gap + training-time mechanism)

Candidate pair. The bib-mapping task provides a direct SDPO-versus-MOPD comparison. On this prompt, the SDPO checkpoint at training step 50 reaches accuracy 0.175, while the MOPD checkpoint at the same step solves it perfectly.

SDPO failure. The SDPO trajectory mixes up the role of the permutation arrays:

```
bib_to_person = [0] * (N + 1)
for i in range(N):
    bib_to_person[Q[i]] = P[i]
result = [str(bib_to_person[Q[i]]) for i in range(N)]
```

This is incorrect in two ways: `P[i]` is the stared-at person, not the wearer of bib `Q[i]`, and the final answer stops in person-ID space instead of projecting back to the target person's bib number. The program preserves the rough shape of the mapping task but composes the wrong relations.

MOPD success. The corresponding MOPD sample keeps the two identifier spaces separate throughout:

```
bib_to_person = [0] * (N + 1)
for i in range(N):
    bib_to_person[Q[i]] = i + 1
for i in range(1, N + 1):
    person = bib_to_person[i]
    target_person = P[person - 1]
    result.append(str(Q[target_person - 1]))
```

This trajectory performs the full chain correctly: `bib` $\rightarrow$ `wearing person` $\rightarrow$ `stared-at person` $\rightarrow$ `target bib`.

Training-time context: MOPD encounters the same failure pattern during rollout. The same bib-mapping prompt appears in the MOPD rollout stream at training step 1 with seven failed peers and one successful peer; the failed peers exhibit the same identifier-space ambiguity that SDPO retains at checkpoint time. Because MOPD injects failed peers directly into the teacher context, the teacher supervising the successful target on this prompt is exposed to a concrete instance of this bug paired with an explicit "avoid this mistake" instruction. SDPO has no equivalent channel.

C.2 Case Study 2: Dropped middle-character constraint

Candidate pair. The evenly spaced `A-B-C` counting task gives a second clean contrast. On this prompt, the SDPO checkpoint at training step 50 reaches accuracy 0.825, while the MOPD checkpoint at the same step is fully correct.

SDPO failure. The SDPO code enforces equal spacing and checks the two endpoints, but drops the requirement that the middle character must be `B`:

```
for j in range(1, n-1):
    for i in range(j):
        if S[i] == 'A':
            k = 2 * j - i
            if k < n and S[k] == 'C':
                count += 1
```

This overcounts invalid triples such as `A-R-C` in the test case `ARC`, where the correct answer is 0. The failure is a local semantic omission: the positional structure is preserved, but one label constraint from the specification is forgotten.

MOPD success. The MOPD sample reinstates the missing check explicitly:

```
for j in range(1, n - 1):
    if s[j] == 'B':
        for i in range(j):
            if s[i] == 'A':
                k = 2 * j - i
                if k < n and s[k] == 'C':
                    count += 1
```

The program is structurally close to the SDPO attempt but preserves the full symbolic constraint set.

Takeaway. Case Studies 1 and 2 together show that MOPD's advantage in avoiding the failure pattern is not tied to a single error type. In Case Study 1, SDPO fails by composing the wrong identifier relations; here it fails by dropping one clause from the specification. MOPD succeeds in both settings; the contrast holds across two distinct error families rather than depending on a single failure mode.

C.3 Case Study 3: Training dynamics — MOPD stabilizes early, SDPO fails late

Candidate pair. The direction-opposite task illustrates the training-time gap. The SDPO checkpoint fails on this prompt at both step 50 and step 100 with accuracy 0.0: the step-50 sample is truncated, and the step-100 sample is flagged for a missing ‘‘‘`python` block. The MOPD checkpoint at step 50 already solves the same prompt with accuracy 1.0.

Early MOPD signal. In the MOPD rollout stream, a correct solution first appears at rollout step 10 with score 1.0—the earliest observed point at which the repaired behavior is present in training-time rollouts for this prompt.

Late SDPO failure. At rollout step 80, SDPO still fails with accuracy 0.0, receiving feedback: `Incorrect Format: Put your code inside a ‘‘‘python ... ‘‘‘ block.`

Takeaway. The gap here is not algorithmic: MOPD internalizes basic output-format discipline by approximately step 10, while SDPO has not stabilized that behavior even late in training.

C.4 Overall Pattern

Taken together, the three case studies in lcv benchmark reveal a consistent picture. The baseline's failures are narrow and consequential—a dropped constraint, a confused identifier space, an unresolved format requirement—rather than wholesale algorithmic breakdowns. These are precisely the errors that peer conditioning is designed to surface: by exposing the teacher to failed peers that exhibit the same bug, MOPD produces a sharper supervision signal at the critical decision points. Case Study 1 compared through training logs; Case Studies 2 and 3 provide corroborating evidence across a different failure type and a training-dynamics lens. The combined evidence supports the conclusion that MOPD's gains arise from a more targeted teacher signal, not from discovering fundamentally different solution strategies.

D Diversity

Distinct-n. Distinct-n [33] is the ratio of unique n-grams to total n-grams across the concatenated token stream of all responses:

$$\text{Distinct-}n = \frac{|\{\text{unique } n\text{-grams}\}|}{\text{total } n\text{-grams}}. \tag{11}$$

We report Distinct-1 and Distinct-2.

AST node Jaccard distance (code only). Each response is parsed into a Python AST and its node-type multiset $\mathbf{a}_i$ is extracted. Pairwise Jaccard distance is

$$d_{ij}^{\mathrm{AST}} = 1 - \frac{\sum_t \min(a_{it},\, a_{jt})}{\sum_t \max(a_{it},\, a_{jt})}, \tag{12}$$

where t ranges over all AST node types present in either response. Responses that fail to parse are excluded; the metric is omitted for a prompt if fewer than two responses parse successfully. The per-prompt score is the mean over all valid pairs.

Code Extraction for Diversity Measurement. For code-focused diversity analysis, we do not compute metrics on the raw response text directly. Instead, each response is first converted into a code view using a simple fenced-code-block extraction rule. Specifically, we match Markdown code blocks of the form ```python ... ``` or ``` ... ``` with a regular expression, and if multiple code blocks appear in one response, we keep only the last matched block. The extracted block is then stripped of leading and trailing whitespace and used as the canonical code content for subsequent diversity computation. If no fenced code block is found, the entire response text is used as a fallback. This procedure is intentionally lightweight: it does not perform full Markdown parsing, and only the `python` language tag is explicitly removed. As a result, responses with other language tags may retain the tag text inside the extracted content.

Results. Table 8 summarises the diversity metrics across the three candidate sets evaluated on 131 shared prompts with up to eight rollouts each.

Table 8: Diversity metrics (mean over prompts, n=131). Higher is better for all columns. **Bold** marks the best value per column.

Method	Distinct-1	Distinct-2	AST Jaccard dist.
MOPD	**0.052**	**0.141**	**0.129**
SDPO	0.027	0.110	0.110
Baseline	0.025	0.111	0.092

MOPD produces rollouts with substantially higher lexical and structural diversity: Distinct-1 is roughly $2\times$ that of SDPO and the baseline, and the mean pairwise AST node-type Jaccard distance is 17% higher than SDPO.

E More Training Details and Training Statistics

Compute Resources. All self-distillation experiments are conducted on a single node of 8×H100 GPUs and a single node of 8×A100 GPUs. Teacher–student distillation experiments are conducted on 2 nodes each equipped with 8×H100 GPUs.

Table 9: Per-step teacher-query cost comparison. Teacher context length is measured in tokens (mean over training steps). Wall-clock time is measured per gradient update step on 8×H100 GPUs with batch size 32 and 8 rollouts per prompt.

Dataset	Method	Time Gen	Time Reward	Time Adv	Time Update	Total
LCB	SDPO	37	400	1	78	550
	MOPD	37	400	2	90	600
QA-Chemistry	SDPO	5	0.1	1	75	90
	MOPD	5	0.1	2	110	120

Time Cost. We report several wall-clock timing metrics in seconds to break down the cost of each training iteration. To avoid startup overhead, we report the average timing measured after the first five training steps, since earlier steps may include additional one-time loading and initialization

costs. `timing_s/gen` measures the time spent generating sampled responses from the current policy. `timing_s/reward` measures the time required to obtain training rewards, including reward-model scoring or task-specific reward computation. `timing_s/update_actor` measures the wall-clock time of the actor optimization stage. In the SDPO setting, this stage includes the student forward pass, the teacher forward pass used to produce teacher logits, the self-distillation loss computation, and the subsequent backward pass and optimizer update. Finally, `timing_s/adv` measures the time spent in the advantage-processing stage, which includes reward post-processing and the computation of training advantages used for policy optimization. `timing_s/step` measures the total duration of one training step, covering the full pipeline from rollout generation to optimization. The reported stage times do not sum exactly to the full step time because they only cover the main timed components of the training loop. The total step time also includes additional overhead such as data movement, batch reorganization, mask construction, metric logging, coordination between workers, and other small bookkeeping operations that are not exposed as separate timing entries. As a result, the difference between the sum of the listed stage times and the total step time can be understood as uncategorized system and orchestration overhead.

Table 9 reports the per-step wall-clock time breakdown for SDPO and MOPD across two datasets. The dominant cost in both methods is reward computation and parameter update; rollout generation and advantage estimation are comparatively negligible. MOPD introduces additional overhead primarily in the update phase, as the teacher must process longer peer-conditioned contexts. Despite this, the total per-step time increases modestly: from 550s to 600s on LCB **(a 9.1% increase)** and from 90s to 120s on QA-Chemistry **(a 33.3% increase)**. These results indicate that the computational overhead of peer conditioning is small relative to the overall training cost, and that MOPD's consistent performance gains are not simply an artifact of increased compute budget.

Table 10: Science QA results with standard deviation over 3 runs. Bold marks the best result in each column.

Method	Biology	Chemistry	Physics	Materials
SDPO	$50.60_{\pm 2.1}$	$73.22_{\pm 2.5}$	$67.36_{\pm 1.9}$	$72.34_{\pm 2.6}$
MOPD	$55.69_{\pm 1.6}$	$74.29_{\pm 2.0}$	$76.25_{\pm 2.8}$	$78.59_{\pm 1.9}$

Training stability. Table 10 reports the mean and standard deviation of MOPD across 3 independent runs on the Science QA benchmarks. While MOPD consistently achieves the best performance across all four domains, the standard deviations reveal moderate variance, particularly in Physics (± 2.8) and Chemistry (± 2.0), suggesting that training is not fully stabilized across runs. We attribute this to the stochastic nature of on-policy rollout sampling and the sensitivity of peer-context construction to the specific success/failure partition observed in each run. Improving training stability through more robust peer selection or variance reduction techniques remains an avenue for future work.

F More Related Work

On-Policy Distillation of Large Language Models Knowledge distillation classically transfers a teacher's predictive distribution to a smaller student through supervision on teacher- or human-curated sequences [20, 29, 62]. For autoregressive language models, this off-policy regime exhibits a well-known limitation: the student is trained on states it never visits at inference, and prediction errors compound over long generations [16, 44, 52]. The central response is to move the supervision signal onto trajectories drawn from the student's own evolving policy, anchoring training on the distribution that will be operated over at deployment [1, 14, 32, 31]. Within this framework, a substantial body of work focuses on how to construct and weight the per-token learning signal. Forward-, reverse-, and skewed-divergence objectives offer different trade-offs between mode coverage and mode seeking [14, 32, 56, 58, 27]. Other works adapt the supervision to vocabulary mismatches between teacher and student [3], reweight tokens by uncertainty or salience [22, 73], or interpolate between fully on- and off-policy regimes via teacher-prefixed rollouts [61, 72]. Black-box variants drop the teacher-logit requirement and supervise the student through sampled outputs, sequence-level rewards, or adversarial discrimination [67, 25]. The paradigm now underlies widely deployed post-training pipelines for instruction-following, code generation, and mathematical reasoning [64]. A complementary thread documents stability and failure modes of these objectives, including capacity-induced learnability

gaps, divergence pathologies, and entropy collapse, and proposes adaptive losses or curricula to mitigate them [4, 34, 26]. Despite this breadth, the dominant assumption is that the teacher signal for each rollout is constructed independently, leaving cross-rollout structure within a sampling group unexploited.

Self-Distillation and Self-Improvement for Reasoning A second line of work treats the same model as both teacher and student, exploiting either privileged conditioning or repeated sampling to manufacture supervision without an external teacher. Iterative self-training procedures generate candidate solutions, filter by verification or reward, and fine-tune on the surviving trajectories, yielding a coarse form of self-distillation for reasoning and alignment [70, 17, 50, 7, 66]. Subsequent work develops soft self-distillation: one copy of the model is conditioned on additional context—ground-truth solutions, hints, demonstrations, or environment feedback—and produces per-token targets for a second copy that lacks that conditioning [68, 78, 23, 49, 42, 10, 65]. Self-distillation has been particularly active in long chain-of-thought reasoning, where supervision is sparse and individual traces are expensive. One direction compresses or reshapes long reasoning traces during distillation [45, 30]. Another aggregates information across multiple sampled solutions through consistency, voting, or committee teachers, transforming an ensemble of attempts into a richer learning signal [54, 40, 36, 51]. A third direction interleaves rollouts with structured self-critique or search before constructing supervision, expanding the teacher's effective view of the problem [71, 2, 74]. Recent analyses also document the limitations of these procedures, noting that self-distillation can degrade reasoning when the student is asked to imitate signals beyond its competence or when the privileged context misaligns with what the student can reproduce [28]. Across these directions, the common thread is that a single model can sharpen its own training signal when given structured access to additional information or to alternative attempts; whether and how those alternative attempts should be jointly conditioned on within a single distillation step remains underexplored.

Multi-Rollout Reinforcement Learning and Verifier-Guided Improvement A third line of related work concerns post-training paradigms that exploit groups of rollouts and external verification rather than per-token teacher matching. Reinforcement learning from human or AI feedback aligns model behavior with preference labels collected over pairs or groups of completions, with subsequent advances exploring offline, reference-free, and online variants [41, 43, 21, 39, 5]. Reinforcement learning with verifiable rewards extends this idea by replacing learned reward models with programmatic checkers—answer matching, unit tests, type-checkers—and updating the policy from group-relative advantages computed over multiple sampled rollouts per prompt [48, 55, 57, 69, 13, 9]. A unified perspective on off- and on-policy post-training places these methods alongside on-policy distillation and clarifies when each provides the most useful supervision signal [77, 63, 35]. Process- and outcome-level verifiers refine the supervision beyond a single sparse trajectory reward. Step-level verifiers and process reward models grade intermediate states, providing denser credit assignment for reasoning [37, 8, 47, 76, 15, 11]. Hybrid post-training pipelines couple on-policy distillation with reinforcement learning from verifiers, using teacher-derived signals as an auxiliary objective alongside reward optimization [60, 59, 38, 75]. While these methods do leverage multiple rollouts per prompt—typically through advantage normalization, rejection sampling, or success-conditioned filtering—the supervisory signal for any single trajectory is computed without conditioning on the contents of the others. The work most adjacent to ours therefore treats successes and failures from the same group as complementary streams of evidence at the level of advantages or filters, but stops short of routing both into a single, peer-conditioned distillation target.

G Limitation

MOPD relies on the availability of multiple rollouts and a verifier or reward function that can partition them into successful and failed trajectories. This makes the method most directly applicable to verifiable domains such as code, math, tool use, and structured QA, while extending it to open-ended generation tasks may require reliable preference models or process-level evaluators. Peer conditioning increases context length and teacher-query cost when raw rollouts are concatenated, especially for long reasoning traces; selecting the most informative peers under context-length constraints is an open question. Finally, the quality of the teacher signal depends on the correctness of the success/failure labels: noisy verifiers or mislabeled rollouts may introduce misleading positive

or negative evidence. Future work should study more robust peer selection, better handling of noisy feedback, and extensions to less easily verifiable tasks.

H Societal Impacts

MOPD may have positive societal impacts by improving the efficiency of post-training and making stronger reasoning capabilities more accessible in smaller language models. Better distillation can reduce inference cost and energy use, and may help deploy capable models in resource-constrained settings. In domains such as code generation, tool use, scientific question answering, and mathematical reasoning, more reliable small models could support education, research assistance, and productivity tools.

At the same time, improving the reasoning and tool-use capabilities of smaller models may also increase risks. More capable and cheaper models could be misused for generating harmful code, automating deceptive content, or executing tool-use workflows in unintended ways. Because MOPD learns from both successful and failed trajectories, poor verifier design or biased feedback could also reinforce undesirable behavior if incorrect success/failure labels are treated as reliable supervision. These risks suggest that peer-conditioned distillation should be paired with careful verifier design, safety evaluations, misuse monitoring, and appropriate deployment safeguards, especially when applied to open-ended, high-impact, or tool-using systems.

I Declaration of LLM Usage

We used large language models as assistance tools during the preparation of this paper. Specifically, LLMs were used to help draft, revise, and polish parts of the manuscript, including improving clarity, grammar, and organization. We also used LLM assistance for code writing and debugging, such as drafting utility scripts, checking implementation details, and improving code readability.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Yes, the abstract and introduction accurately reflect the paper's contributions and scope.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: In the appendix.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: The paper does not include theoretical results.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We provide implementation details in the exp section and provide code.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Yes, we provide the code and all the datasets are publicly available.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: Yes, in the exp section.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We have in appendix.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: In the appendix.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: Yes.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: In the appendix.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: the paper poses no such risks.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: Yes

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: we add in the code repo.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: he paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: the paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: in the appendix.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.