Restoring the Sweet Spot: Pass-Rate Weighted Self-Distillation for LLM Reasoning

Anonymous ACL submission

Abstract

Self-Distillation Policy Optimization (SDPO) provides dense token-level credit assignment for reinforcement learning with large language models by leveraging the model's own feedback-conditioned predictions as a self-teacher. Unlike GRPO, however, whose group-relative advantage naturally concentrates learning on a *sweet spot* of intermediate-difficulty questions, SDPO's KL-based advantage lacks an implicit notion of difficulty awareness.

We analyze this gap through the lens of GRPO's advantage normalization. Extending the learnability framework to normalized rewards, we show that normalization absorbs the variance term $p(1-p)$, equalizing leading-order learnability across questions and leaving $\sqrt{p(1-p)}$ as the sole residual scaling factor in the per-question gradient. This analysis yields a simple prescription: weight each question's SDPO loss by $[\hat{p}(1-\hat{p})]^{1/2}$, resulting in SC-SDPO, a scale-consistent variant of SDPO.

The proposed weights are obtained as a zero-cost byproduct of on-policy rollouts with batch-adaptive normalization, inducing an implicit curriculum that dynamically tracks the model's evolving competence. Experiments on scientific reasoning and tool-use benchmarks demonstrate that SC-SDPO consistently improves over SDPO, yielding gains of +3.2/+4.3 (mean@16/maj@16) on Qwen3-8B and +1.8/+3.0 on OLMo-3-7B, while preserving stable training dynamics throughout optimization. Code is available at `https://anonymous.4open.science/r/scsdpo-E686/`.

1 Introduction

Large language models (LLMs) are increasingly post-trained with reinforcement learning in verifiable domains such as code and mathematics (Guo et al., 2025; Jaech et al., 2024; Team et al., 2025; Olmo et al., 2025). The dominant approach, reinforcement learning with verifiable rewards (RLVR), trains models using sparse binary outcome rewards. More recently, on-policy distillation (OPD) has emerged as a complementary paradigm that provides dense, token-level supervision from a teacher model, achieving strong performance on reasoning-intensive tasks (DeepSeek-AI, 2026; Xiao et al., 2026; Yang et al., 2025; Zeng et al., 2026). Despite its effectiveness, RLVR suffers from a fundamental credit-assignment bottleneck: a single scalar reward per attempt indicates whether the model was wrong, but not where or why (Hübotter et al., 2026).

Self-Distillation Policy Optimization (SDPO; Hübotter et al., 2026) offers an elegant solution to this bottleneck. By conditioning the same model on privileged information—such as runtime errors or previously discovered solutions—SDPO constructs a self-teacher that provides dense, logit-level supervision without requiring an external model. This mechanism converts tokenized feedback into fine-grained credit assignment and substantially outperforms Group Relative Policy Optimization (GRPO) (Guo et al., 2025; Shao et al., 2024).

However, while SDPO resolves GRPO's credit-assignment bottleneck, it also sacrifices a useful property that GRPO possesses by construction. In GRPO, the group-relative advantage naturally concentrates learning on a *sweet spot* of intermediate-difficulty questions, where the model sometimes succeeds and sometimes fails. By contrast, SDPO's KL-based distillation loss does not implicitly encode difficulty awareness: its gradient budget is largely insensitive to pass rate, diluting the learning signal precisely where self-distillation is most informative.

Prior work has formalized the role of difficulty in RLVR. Bae et al. (2026) show, under raw binary rewards, that policy improvement is lower-bounded by the reward variance $p(1-p)$. However, this analysis does not account for GRPO's advantage

normalization. We extend their framework to the normalized setting and show that normalization absorbs the $p(1-p)$ factor, equalizing leading-order learnability across questions. The residual per-question gradient magnitude instead scales as $\sqrt{p(1-p)}$ rather than $p(1-p)$ (section 4).

This analysis directly motivates SC-SDPO, which restores the sweet spot to SDPO by scaling each question's loss contribution by $w = [\hat{p}(1-\hat{p})]^{1/2}$. The weighting automatically suppresses all-correct and all-incorrect groups while peaking at $\hat{p} = 0.5$. The weights are recomputed from the current policy's rollouts at every training step using batch-adaptive normalization, forming an implicit curriculum that tracks the model's evolving competence at zero additional rollout cost.

We evaluate SC-SDPO on scientific reasoning and tool-use tasks. Our main findings are:

- SC-SDPO improves over SDPO by +3.2/+4.3 (mean@16/maj@16) on Qwen3-8B and +1.8/+3.0 on OLMo-3-7B across five tasks, demonstrating that difficulty-aware weighting and dense credit assignment are complementary.
- The $\alpha = 1/2$ exponent consistently outperforms $\alpha = 1$, supporting the scale-consistent argument.
- Batch-adaptive weighting outperforms static single-pass schemes while incurring zero additional rollout cost.

2 Related Work

On-policy self-distillation. On-policy distillation trains a student model on its own generations to match a teacher's distribution (Agarwal et al., 2024; Gu et al., 2024; Lu and Lab, 2025; Song and Zheng, 2026). When no external teacher is available, self-distillation methods condition the model on privileged in-context information to construct a self-teacher. Representative approaches include SDPO (Hübotter et al., 2026), SDFT (Shenfeld et al., 2026), OPSD (Zhao et al., 2026), OPSDC (Sang et al., 2026), and G-OPD (Yang et al., 2026b). Several works further improve sample routing within this paradigm: SCOPE (Zheng et al., 2026) assesses signal reliability to adaptively filter noisy teacher corrections and reinforce diverse reasoning paths, HDPO (Ding, 2026) tackles sparse rewards on excessively hard tasks in GRPO by injecting privileged information to construct high-quality distillation targets, SRPO (Li et al., 2026) routes samples based on correctness, applying token-level distillation to errors and GRPO to successes, TRACE (Wang et al., 2026) introduces a token-level routing framework that adaptively splits the training objective between self-distillation and GRPO, RLSD (Yang et al., 2026a) addresses privileged information leakage by decoupling the update direction (determined by environment reward) from the update magnitude (modulated by a per-token teacher-student likelihood ratio), and SD-ZERO (He et al., 2026) takes a two-phase approach, first training the model to revise its own outputs conditioned on binary rewards, then distilling the revision capability back into the generator via on-policy KL minimization.

Difficulty-aware training. DAPO (Yu et al., 2026) discards homogeneous rollout groups in RLVR, and Bae et al. (2026) provide a theoretical foundation under raw binary rewards by showing that policy improvement is lower-bounded by the reward variance $p(1-p)$. GRPO-LEAD (Zhang and Zuo, 2025) and MathForge (Dai et al., 2026) reweight advantages by the difficulty within the RLVR framework. In the on-policy distillation setting, PACED (Xu et al., 2026) derives a Beta-kernel weight $w(p) = p^{\alpha}(1-p)^{\beta}$ from the signal-to-noise ratio structure of distillation gradients, and demonstrates its effectiveness in both teacher-forced distillation and on-policy self-distillation. Our work differs from PACED in two key respects: we adopt $\alpha = \beta = 1/2$ rather than $\alpha = \beta = 1$, justified by section 4; and we compute weights on-policy at every step as a free byproduct of the existing rollout loop, whereas PACED estimates pass rates in a separate offline rollout pass.

3 Preliminaries

3.1 Self-Distillation Policy Optimization

We build upon Self-Distillation Policy Optimization (SDPO; Hübotter et al., 2026), which leverages the in-context learning ability of a language model for credit assignment in reinforcement learning. We briefly review its key components below.

Self-Teacher. Given a question x, the policy π_θ generates a response $y \sim \pi_\theta(\cdot \mid x)$ and receives tokenized feedback f from the environment (e.g., correct solutions). SDPO introduces a *self-teacher* $\pi_\theta(\cdot \mid x, f)$, which is the same policy additionally prompted with the feedback f. Since the self-

teacher observes this extra information in context, it can retrospectively identify mistakes in the student's original attempt y and adjust its next-token predictions accordingly. Crucially, compared with On-Policy Distillation(OPD; Agarwal et al., 2024), this mechanism requires no external teacher model.

Training Objective. SDPO minimizes a KL-divergence-based distillation loss between the student and the self-teacher at each token position t:

$$\mathcal{L}_{\text{SDPO}}(\theta) := \sum_t \mathrm{KL}\big(\pi_\theta(\cdot \mid x, y_{<t}) \,\|\, \mathrm{sg}(\pi_\theta(\cdot \mid x, f, y_{<t}))\big), \tag{1}$$

where $\mathrm{sg}(\cdot)$ denotes the stop-gradient operator, which prevents gradients from flowing through the teacher.

Dense Credit Assignment. The gradient of $\mathcal{L}_{\text{SDPO}}$ can be interpreted as a policy gradient with per-token, per-logit advantages. Under the reverse KL formulation, the advantage takes the form:

$$A_{i,t}^{\text{SDPO}}(\hat{y}_{i,t}) = \log \frac{\pi_\theta(\hat{y}_{i,t} \mid x, f_i, y_{i,<t})}{\pi_\theta(\hat{y}_{i,t} \mid x, y_{i,<t})}, \tag{2}$$

where $\hat{y}_{i,t}$ ranges over the vocabulary at position t. This contrasts with GRPO (Shao et al., 2024; Guo et al., 2025), whose advantage is constant across all tokens within a rollout. In practice, the KL divergence is approximated by retaining only the top-K logits of the student distribution to reduce memory overhead. Additionally, the symmetric Jensen–Shannon divergence can replace the KL divergence in Eq. (1) to further stabilize training. When the symmetric Jensen–Shannon divergence replaces the KL in Eq. (1), the per-token advantage is given as:

$$A_{\text{JSD}}(\hat{y}_{i,t}) = \frac{1}{2} \log \frac{\pi_\theta(\hat{y}_{i,t})}{M(\hat{y}_{i,t})}, \tag{3}$$

where $M(\hat{y}_{i,t}) = \frac{1}{2}(\pi_\theta(\hat{y}_{i,t}) + \pi_\theta(\hat{y}_{i,t}|f))$ is the mixture. The derivation is provided in section A.3.

3.2 GRPO and the Reward Sweet Spot

In GRPO (Guo et al., 2025; Shao et al., 2024), for each question x, a group of G rollouts $\{y_i\}_{i=1}^G$ is sampled from the current policy, and each receives a binary reward $r_i \in \{0, 1\}$. The per-rollout estimated advantage is computed by normalizing the reward relative to the group:

$$\hat{A}_i^{\text{GRPO}} = \frac{r_i - \bar{r}}{\mathrm{std}(\{r_i\})}, \tag{4}$$

where $\bar{r} = \frac{1}{G}\sum_{i=1}^G r_i$ is the group mean. When all rollouts succeed ($\bar{r} = 1$) or all fail ($\bar{r} = 0$), the advantages collapse to zero and no learning occurs. Dai et al. (2026) formalize the total update magnitude for a single question as (Theorem 1):

$$\sum_{i=1}^G |\hat{A}_i^{\text{GRPO}}| = \sum_{i=1}^G \left| \frac{r_i - \bar{r}}{\mathrm{std}(\{r_i\})} \right| = 2G\sqrt{p(1-p)}. \tag{5}$$

Bae et al. (2026) formalize the learnability bound under the raw (unnormalized) binary reward $r_{\text{acc}} \in \{0, 1\}$, proving that the expected policy improvement $J(\theta') - J(\theta)$ is lower-bounded by the within-group reward variance:

$$J(\theta') - J(\theta) \;\geq\; \frac{1}{2\beta^2} \cdot \mathrm{Var}[r_{\text{acc}}] = \frac{1}{2\beta^2} \cdot p(1-p), \tag{6}$$

where $\beta > 0$ is the KL regularization temperature. Since $p(1-p)$ is a concave quadratic that attains its unique maximum at $p = 0.5$, the learning signal is strongest when roughly half the rollouts are correct, defining a *reward sweet spot* at intermediate difficulty. We note that this conclusion rests on the raw reward prior to any normalization. As we show in section 4, GRPO's group-relative normalization equalizes the leading-order learnability across all non-degenerate questions, broadening the sweet spot from a single point at $p = 0.5$ to the entire interval $p \in (0, 1)$; the residual p-dependence in the per-step gradient magnitude is $\sqrt{p(1-p)}$ rather than $p(1-p)$.

4 Gap Between SDPO and GRPO

The two frameworks offer complementary strengths that can be seen by comparing their per-question gradient contributions. Let $\{x_j\}_{j=1}^M$ be the questions in a training batch, with $\mathcal{G}_j$ denoting the set of rollouts sampled from π_θ for question x_j. Both GRPO and SDPO can be expressed in a unified policy gradient form:

$$\nabla_\theta \mathcal{L}_j = \sum_{i \in \mathcal{G}_j} \sum_t A_{i,t} \cdot \nabla_\theta \log \pi_\theta(y_{i,t} \mid x_j, y_{i,<t}). \tag{7}$$

We decompose the advantage into two complementary factors—a *question-level* component A_j^{q} reflecting difficulty awareness, and a *token-level* component $A_{i,t}^{\text{tok}}$ providing dense credit assignment:

$$A_{i,t} = A_j^{\text{q}} \cdot A_{i,t}^{\text{tok}}. \tag{8}$$

Under this decomposition, the two methods each capture only one factor:

$$\text{GRPO:} \quad A^{\mathrm{q}}_j = g(\hat{p}_j), \quad A^{\mathrm{tok}}_{i,t} = 1, \tag{9}$$

$$\text{SDPO:} \quad A^{\mathrm{q}}_j = 1, \quad A^{\mathrm{tok}}_{i,t} = \log \frac{\pi_\theta(\hat{y}_{i,t} \mid f_i)}{\pi_\theta(\hat{y}_{i,t})}. \tag{10}$$

where $g(\hat{p}_j)$ denotes the effective question-level signal strength of GRPO, and we suppress the shared conditioning on $(x, y_{i,<t})$ in $A^{\mathrm{tok}}_{i,t}$ for brevity.

What each method lacks. GRPO's group-relative advantage $\frac{r_i - \bar{r}}{\mathrm{std}(\{r_i\})}$ implicitly modulates the question-level signal—its aggregate magnitude scales as $\sqrt{\hat{p}_j(1-\hat{p}_j)}$, peaking at $\hat{p}_j = 0.5$ and vanishing for all-correct or all-incorrect groups—but assigns a uniform advantage across all tokens within a rollout ($A^{\mathrm{tok}}_{i,t} = 1$). SDPO provides dense, per-token credit assignment through the self-teacher log-ratio but treats all questions equally ($A^{\mathrm{q}}_j = 1$), losing the difficulty-aware modulation that GRPO possesses by construction. We verify this empirically in Figure 4 (Appendix): the per-token SDPO advantage magnitude remains largely constant across pass rates, confirming that SDPO's distillation loss does not implicitly recover question-level signal modulation.

Why the ideal scales as $\sqrt{p(1-p)}$, not $p(1-p)$. The variance-based scaling $p(1-p)$ originates from the analysis of Bae et al. (2026), who derive their lower bound (Eq. 6) under the raw binary reward $r_{\mathrm{acc}} \in \{0, 1\}$ without accounting for advantage normalization. However, GRPO does not use the raw reward as its advantage; instead, it applies group-relative normalization (Eq. 4), yielding the standardized advantage $\hat{A}_i = (r_i - \bar{r})/\mathrm{std}(\{r_i\})$.

We extend the theoretical framework of Bae et al. (2026) to this normalized setting. Specifically, we apply their cumulant generating function (CGF) analysis (Proposition 6.1 in Bae et al. 2026) to the standardized reward $\tilde{r} := (r-p)/\sqrt{p(1-p)}$. The main result (Proposition 1 in section A.5) shows that the leading term of the learnability bound becomes a constant $1/(2\beta^2)$ independent of p:

$$D_{\mathrm{KL}}(\pi_{\mathrm{init}} \,\|\, \tilde{\pi}^*) = \frac{1}{2\beta^2} + O(\beta^{-3}), \tag{11}$$

where $\tilde{\pi}^*$ is the optimal policy under the normalized reward. The normalization absorbs the $p(1-p)$ factor from the raw-reward bound, equalizing leading-order learnability across all non-degenerate questions. The only remaining p-dependence is GRPO's per-question gradient magnitude $\sqrt{p(1-p)}$ (Eq. 5), which the ideal objective adopts as its question-level factor.

An ideal objective. Given that the natural question-level scaling is $\sqrt{p(1-p)}$, the decomposition in Eq. (8) suggests a natural ideal objective that combines the strengths of both frameworks: dense token-level credit assignment from SDPO together with GRPO's question-level difficulty modulation. Concretely, the ideal per-question gradient takes the form:

$$\nabla_\theta \mathcal{L}^{\mathrm{ideal}}_j = \underbrace{[\hat{p}_j(1-\hat{p}_j)]^{1/2}}_{A^{q,\star}_j\text{: question-level}} \cdot \underbrace{\sum_{i \in \mathcal{G}_j} \sum_t A^{\mathrm{SDPO}}_{i,t} \cdot \nabla_\theta \log \pi_\theta(y_{i,t} \mid x_j, y_{i,<t})}_{\text{token-level (SDPO)}}, \tag{12}$$

where the question-level factor $A^{q,\star}_j = [\hat{p}_j(1-\hat{p}_j)]^{1/2}$ reproduces the natural scaling that GRPO obtains implicitly through group-relative advantage normalization.

5 Method

The ideal objective (Eq. 12) prescribes a specific question-level factor but leaves open how to implement it within SDPO's existing training loop. We show that a single scalar modification suffices. As shown in Eq. (8), the advantage decomposes into a question-level factor A^{q}_j and a token-level factor $A^{\mathrm{tok}}_{i,t}$. We propose to set $A^{\mathrm{q}}_j = \bar{w}_j$, a per-question weight derived from on-policy pass rates, yielding the Scale-Consistent SDPO (SC-SDPO) gradient:

$$\nabla_\theta \mathcal{L}_j = \sum_{i \in \mathcal{G}_j} \sum_t \bar{w}_j \cdot A^{\mathrm{SDPO}}_{i,t} \cdot \nabla_\theta \log \pi_\theta(y_{i,t} \mid x_j, y_{i,<t}). \tag{13}$$

The modification is minimal—a single scalar multiplier per question—yet it restores the question-level signal modulation that SDPO lacks. We detail the construction of $\bar{w}_j$ in two parts: the weighting function (section 5.1), and the batch-adaptive normalization (section 5.2).

5.1 Weighted SDPO Objective

Of the $|\mathcal{G}_j|$ rollouts, let k_j succeed, giving an empirical pass rate $\hat{p}_j = k_j/|\mathcal{G}_j|$. The SC-SDPO objective is:

$$\mathcal{L}_{\text{SC-SDPO}}(\theta) := \sum_{j=1}^{M} \bar{w}_j \sum_{i \in \mathcal{G}_j} \sum_{t} \mathrm{KL}\big(\pi_\theta(\cdot \mid x_j, y_{i,<t}) \,\big\|\, \mathrm{sg}\big(\pi_\theta(\cdot \mid x_j, f_i, y_{i,<t})\big)\big), \tag{14}$$

where $\bar{w}_j$ is a normalized, question-level weight derived from $\hat{p}_j$. The inner two sums are identical to the original SDPO loss; the only change is the outer multiplication by $\bar{w}_j$.

The unnormalized weight takes the form:

$$w_j = \big[\hat{p}_j(1-\hat{p}_j)\big]^{\alpha}, \quad \alpha > 0. \tag{15}$$

The term $\hat{p}(1-\hat{p})$ is the Bernoulli variance of the rollout outcomes and equals zero when $\hat{p} \in \{0, 1\}$, reaching its maximum at $\hat{p} = 0.5$. Questions where the policy sometimes succeeds and sometimes fails—the regime where contrastive self-distillation is most informative—receive the largest weight. The exponent α controls how sharply the weight concentrates around this sweet spot. We set $\alpha = 1/2$ by default, as justified by the scale-consistency analysis in section 4.

5.2 Batch-Adaptive Normalization

As the policy evolves and more questions migrate to the extremes ($\hat{p} \to 1$), the number of non-degenerate questions $|\mathcal{S}|$ and the raw weight w_j sum both change, potentially destabilizing the effective learning rate. We apply a simple within-batch normalization. Let $\mathcal{S} = \{j : w_j > 0\}$ be the set of questions with non-degenerate pass rates ($\hat{p}_j \notin \{0, 1\}$). The normalized weight is:

$$\bar{w}_j = \begin{cases} \dfrac{w_j}{\frac{1}{|\mathcal{S}|}\sum_{l \in \mathcal{S}} w_l}, & \text{if } j \in \mathcal{S}, \\ 0, & \text{otherwise.} \end{cases} \tag{16}$$

This ensures that the mean weight over informative questions is always 1, preventing the effective learning rate from drifting as the batch composition changes during training. It does not affect the relative weight distribution and hence does not alter the scale-consistency to the ideal objective.

Because SDPO is an on-policy algorithm that already samples G rollouts per question and evaluates them at every training step, the pass rate $\hat{p}_j$ is a *free byproduct* of the existing pipeline. Computing w_j and $\bar{w}_j$ adds $\mathcal{O}(M)$ scalar operations—negligible compared to the forward and backward passes. Unlike single-pass estimation methods such as PACED (Xu et al., 2026), which compute pass rates once before training and freeze them, our weights are recomputed at every step and therefore automatically track the evolving policy: a previously unsolvable question that enters the sweet spot as training progresses will receive increasing weight, while mastered questions are naturally phased out. This yields an implicit curriculum without manual scheduling.

```
# --- SC-SDPO: 6-line modification to SDPO ---
# Input: rewards (B,), uids (B,), alpha=0.5

# Step 1: per-question pass rate
for uid in unique(uids):
    p[uid] = rewards[uids == uid].mean()

# Step 2: std-scaling weight
w_raw = {uid: (p * (1 - p)) ** alpha
     for uid, p in p.items()}

# Step 3: batch-adaptive normalization
w_nz = [w for w in w_raw.values() if w > 0]
w = {uid: w / mean(w_nz) if w > 0 else 0
     for uid, w in w_raw.items()}

# Step 4: apply to SDPO loss
per_token_loss = per_token_loss * w[uids]
```

Figure 1: Implementation of SC-SDPO. The entire modification reduces to computing a per-question scalar weight from on-policy rollouts (Steps 1–3) and a single element-wise multiplication on the existing SDPO loss (Step 4). No architectural changes, additional rollouts, or hyperparameter tuning beyond the default $\alpha = 0.5$ are required.

Summary. SC-SDPO modifies SDPO by a single per-question scalar multiplication. The weight $\bar{w}_j$ restores the question-level gradient modulation that GRPO obtains implicitly, concentrating training on the difficulty sweet spot while preserving SDPO's dense, per-token credit assignment. The implementation is shown in Figure 1.

6 Experiment

6.1 Experimental Setting

Models and training. Following Hübotter et al. (2026), we conduct experiments on Qwen3-8B (Yang et al., 2025) and OLmo3-7B-Instruct (Olmo et al., 2025), both in non-thinking mode. Qwen3-8B serves as our primary model;

OLmo3-7B-Instruct is used to verify generalization across model families. For OLMo-3-7B, we report only the core comparison between SDPO and SC-SDPO ($\alpha = 0.5$) due to computational constraints; the full ablation suite (PACED, Hard Filter, SRPO, $\alpha = 1$) is conducted exclusively on Qwen3-8B. All runs are trained for 200 steps on 2 NVIDIA H200 GPUs using the verl framework (Sheng et al., 2024). We report results in terms of training steps rather than wall-clock time (as in Hübotter et al., 2026; Li et al., 2026), since step-based evaluation is independent of hardware configuration and ensures reproducibility across different compute environments.

Data. We evaluate on two task families for which neither model has been explicitly fine-tuned: (1) **Science Q&A** (Chemistry, Physics, Biology, Materials Science), using the undergraduate-level reasoning subsets (L3) from SciKnowEval (Feng et al., 2024); and (2) **Tool Use**, mapping tool-API specifications and user requests to correct tool calls, using ToolAlpaca (Tang et al., 2023). We adopt the same train-test splits as Hübotter et al. (2026). Prompt templates are provided in section A.7.

Methods. We compare 8 configurations: (1) **GRPO**: the standard group-relative policy optimization baseline, using the same optimization setup as Hübotter et al. (2026); (2) **GRPO w/o Norm**: GRPO without the standard-deviation normalization in the advantage, using raw centered rewards $r_i - \bar{r}$ directly; (3) **SDPO**: the unweighted baseline; (4) **SDPO w/ PACED**: pass-rate weights $w_j = \hat{p}_j(1-\hat{p}_j)$ estimated in a single offline rollout pass before training and frozen throughout, following Xu et al. (2026); (5) **SDPO w/ Hard Filter**: a binary filtering strategy that retains only questions with $0.2 \leq \hat{p}_j \leq 0.8$ and discards the rest, recomputed on-policy at each step; (6) **SRPO** (Li et al., 2026): uses the sample-level routing and dynamic-weighted SDPO mechanisms proposed in the original paper with default hyperparameters; all other settings (learning rate, batch size, number of rollouts, etc.) are kept consistent with SDPO and the other methods in this work; (7) **SC-SDPO** (α=1): batch-adaptive variance-based weighting $w_j = \hat{p}_j(1 - \hat{p}_j)$ with on-policy recomputation and normalization at every step; (8) **SC-SDPO** (α=0.5): our proposed method, batch-adaptive scale-consistent weighting $w_j = [\hat{p}_j(1 - \hat{p}_j)]^{1/2}$ with on-policy recomputation and normalization at every step. All other hyperparameters (learning rate, batch size, number of rollouts, EMA rate) are held constant across configurations to isolate the effect of the weighting scheme. Details of implementation and hyperparameter are given in section A.2.

6.2 Main Results

Tables 1 and 2 summarize the main results on Qwen3-8B and OLMo-3-7B, respectively.

SC-SDPO achieves the best overall performance. On Qwen3-8B, SC-SDPO (α=0.5) achieves the highest average accuracy on both mean@16 (74.8%) and maj@16 (76.9%), outperforming the SDPO baseline by +3.2 and +4.3 points, respectively. The improvement is consistent across tasks, with particularly large gains on Physics (+5.8/+6.4) and Biology (+5.0/+6.6), where the pass rate distribution offers the broadest intermediate zone for difficulty-aware weighting to exploit (Figure 5a).

Dense credit assignment is essential. Both GRPO variants substantially underperform all SDPO-based methods, confirming that SDPO's dense, token-level credit assignment provides a fundamental advantage over GRPO's uniform per-token signal. Notably, removing the advantage normalization (GRPO w/o Norm) further degrades performance, consistent with our theoretical analysis in section 4: normalization equalizes learnability across questions and is a key component of GRPO's implicit difficulty awareness. These results underscore that SC-SDPO's gains come from combining both ingredients—dense credit and difficulty modulation—rather than from either alone.

scale-consistent weighting outperforms variance-based weighting. Comparing the two exponents, α=0.5 consistently outperforms α=1 across all tasks, with an average gap of +3.5 on mean@16 and +5.2 on maj@16. This validates our theoretical argument (section 4): under GRPO's normalized rewards, the leading-order learnability is equalized across questions, and the residual per-question gradient magnitude scales as $\sqrt{p(1-p)}$; α=1 over-suppresses this signal by an additional factor of $\sqrt{p(1-p)} \leq 0.5$.(Figure 2, Right).

Static weighting and hard filtering underperform. PACED, which applies variance-based weights (α=1) frozen at initialization, is the weakest method—falling 3.4/3.1 points below the unweighted SDPO baseline on average. This

Method	Chem.		Phys.		Bio.		Mat.		Tool Use		Avg.	
	mean	maj	mean	maj	mean	maj	mean	maj	mean	maj	mean	maj
Qwen3-8B	42.4	35.7	57.7	53.8	28.7	14.0	59.7	59.6	58.3	57.4	49.4	44.1
+ GRPO	68.5	70.9	64.7	68.3	58.9	60.0	69.8	69.9	67.3	67.6	65.8	67.3
+ GRPO w/o Norm	63.9	67.0	69.1	76.5	53.2	53.9	73.3	74.5	63.5	63.4	64.6	67.1
+ SDPO	78.7	79.2	75.8	78.5	60.4	61.9	78.0	78.6	65.1	64.8	71.6	72.6
+ SDPO w/ PACED	74.7	75.5	69.2	71.7	57.0	59.0	75.4	75.8	64.5	65.7	68.2	69.5
+ SDPO w/ Hard Filter	80.0	80.4	76.4	79.6	56.0	56.4	77.3	78.1	63.8	64.8	70.7	71.9
+ SRPO	80.5	**81.6**	73.4	76.3	56.0	60.8	77.2	**80.3**	**71.8**	**73.3**	71.8	74.5
+ SC-SDPO (α=1)	80.0	79.5	75.4	77.3	58.1	56.3	77.1	78.3	66.2	66.9	71.3	71.7
+ SC-SDPO (α=0.5)	**80.6**	81.3	**81.6**	**84.9**	**65.4**	**68.5**	**79.3**	**80.3**	67.3	69.3	**74.8**	**76.9**

Table 1: Qwen3-8B results comparing mean@16 and maj@16 (%). **Bold** and underlined values indicate the best and second-best results, respectively.

Method	Chem.		Phys.		Bio.		Mat.		Tool Use		Avg.	
	mean	maj	mean	maj	mean	maj	mean	maj	mean	maj	mean	maj
OLMo-3-7B	22.1	9.5	33.3	30.0	13.4	8.0	36.6	30.9	43.4	45.6	29.7	24.8
+ SDPO	81.1	81.4	68.2	68.7	52.0	52.0	78.0	80.0	62.1	65.0	68.3	69.4
+ SC-SDPO (α=0.5)	**81.5**	**82.5**	**71.1**	**73.4**	**55.9**	**58.2**	**79.1**	**80.7**	**62.9**	**67.3**	**70.1**	**72.4**

Table 2: OLMo-3-7B results comparing mean@16 and maj@16 (%). **Bold** and underlined values indicate the best and second-best results, respectively.

degradation is attributable to both the squared-scale suppression of α=1 and the staleness of frozen weights, which become increasingly misaligned with the model's evolving competence (section A.6). Hard filtering (retaining $0.2 \leq \hat{p} \leq 0.8$) performs closer to SDPO but still trails SC-SDPO by 4.1/5.0 points on average, confirming that soft, continuous weighting extracts more signal than binary thresholding.

Comparison with SRPO. SRPO achieves competitive results, ranking second on maj@16 (74.5%) and obtaining the best scores on Tool Use (71.8/73.3) and Materials maj@16 (80.3). However, SRPO's sample-level routing mechanism introduces substantially higher gradient variance (Figure 2, Right) and training instability compared to SC-SDPO, which achieves a higher average with smoother optimization dynamics. Moreover, SRPO's gains on Tool Use—where pass rates are concentrated at the extremes (section A.6)—suggest that its GRPO branch provides complementary signal in regimes where difficulty-aware weighting has limited leverage.

Cross-model generalization. On OLMo-3-7B (Table 2), SC-SDPO (α=0.5) improves over SDPO by +1.8/+3.0 on mean@16/maj@16, with gains across all five tasks. This confirms that the benefit of pass-rate weighting is not specific to the Qwen model family.

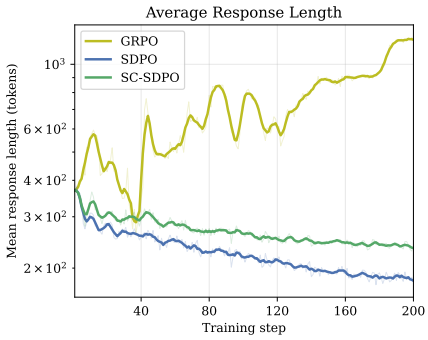


Figure 3: Average response length (in tokens) over 200 training steps, averaged across five datasets.

Response length. Figure 3 shows that SC-SDPO and SDPO exhibit a declining trend in response length as training progresses, consistent with prior findings that on-policy RL encourages more concise reasoning (Hübotter et al., 2026). Notably, SC-SDPO maintains consistently higher response lengths than the SDPO baseline throughout training, suggesting that pass-rate weighting steers the model toward more elaborate reasoning on intermediate-difficulty questions rather than col-

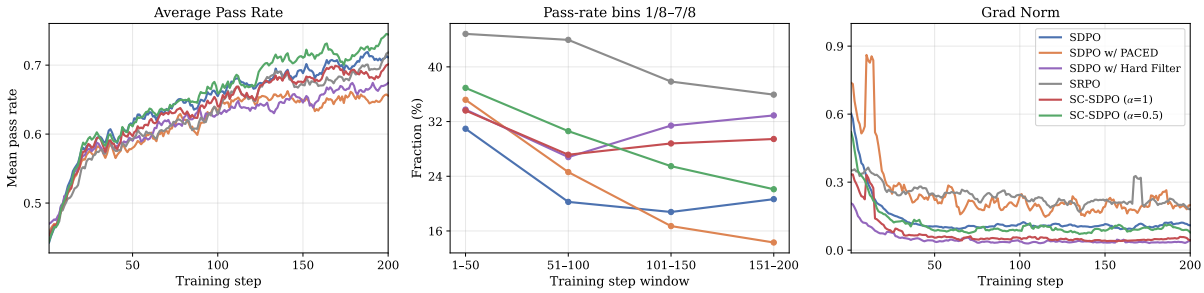


Figure 2: Averaged training dynamics across five datasets. **(Left)** Mean pass rate per training step. **(Middle)** Fraction of training samples in pass-rate bins $\frac{1}{8}$–$\frac{7}{8}$ per 50-step window. **(Right)** Actor gradient norm per training step.

lapsing to short, confident answers on already-mastered ones.

6.3 Training Dynamics Analysis

Figure 2 presents three complementary views of the training dynamics. Per-dataset breakdowns and extended analysis are provided in section A.6.

Pass rate progression (Left). All methods exhibit rising mean pass rates as training progresses, reflecting the model's growing competence. SC-SDPO (α=0.5) achieves the highest mean pass rate in the later stages of training, indicating that concentrating gradient budget on the sweet spot accelerates overall skill acquisition. ToolUse is a notable exception where pass rates remain low across all methods due to the polarized difficulty distribution (cf. Figure 5a in the Appendix).

Sweet spot retention (Middle). SRPO maintains the highest fraction of intermediate-pass-rate samples throughout training, which is expected given its routing design that preserves mixed-outcome groups. However, as shown in Table 1, this does not translate into the best average accuracy, suggesting that a large sweet spot is necessary but not sufficient—optimization stability and signal quality matter equally. Among non-routing methods, SC-SDPO (α=0.5) retains the highest intermediate fraction in early training (steps 1–50), when the learning frontier is broadest and difficulty-aware weighting has the greatest impact. As more questions are mastered, the fraction naturally declines across all methods; SC-SDPO converges smoothly, consistent with the implicit curriculum effect of our batch-adaptive design.

Gradient stability (Right). SC-SDPO (α=0.5) maintains a gradient norm profile closely aligned with the unweighted SDPO baseline, confirming that scale-consistent weighting reshapes the gradient distribution across questions without disrupting the overall optimization scale. This stability follows directly from the batch-adaptive normalization (Eq. 16), which preserves unit-mean weights. In contrast, PACED produces large gradient spikes in early training due to two compounding factors: its global normalization allows batch-level weight sums to fluctuate, and its frozen weights become misaligned with the model's evolving competence. SRPO exhibits persistently unstable gradients, suggesting that its sample-level routing mechanism introduces inconsistent gradient directions across steps. The variance-based weighting (α=1) drives the gradient norm well below the baseline, starving the optimizer of learning signal, while the hard filter yields similarly suppressed gradients due to its reduced effective batch size.

7 Conclusion

We identified a fundamental gap between SDPO and GRPO: SDPO provides dense token-level credit assignment but lacks question-level difficulty awareness, while GRPO possesses the opposite strengths. By extending the learnability analysis of Bae et al. (2026) to GRPO's normalized rewards, we showed that advantage normalization absorbs the $p(1-p)$ factor from the raw-reward bound, leaving $\sqrt{p(1-p)}$ as the natural per-question gradient scaling. This analysis directly prescribed SC-SDPO, which restores difficulty awareness to SDPO through a single scalar weight per question. Experiments on scientific reasoning and tool-use tasks confirmed that SC-SDPO consistently outperforms SDPO and all baselines, with $\alpha = 1/2$ validating the scale-consistent scaling predicted by our theory.

8 Limitations

Dependence on pass-rate granularity. With $G = 8$ rollouts per question, the empirical pass rate $\hat{p}$ takes only nine discrete values $\{0, \frac{1}{8}, \ldots, 1\}$. The Beta-type weight $[\hat{p}(1-\hat{p})]^{1/2}$ smooths over this discretization, but fine-grained difficulty distinctions may be lost. Increasing G improves resolution at the cost of additional generation compute.

Binary reward assumption. Our theoretical justification for $\alpha = 1/2$ relies on the Bernoulli structure of binary rewards. In domains with continuous or partial-credit rewards, the relationship between reward variance and gradient signal strength may differ, and the optimal exponent could change accordingly.

Task scope. We evaluate on scientific reasoning and tool-use tasks with verifiable answers. Extending pass-rate weighting to open-ended generation tasks (e.g., creative writing, summarization) would require a proxy competence signal such as reward-model scores, introducing additional noise and calibration challenges.

Model scale. Our experiments are conducted on 7–8B parameter models. While the method is architecture-agnostic, its effectiveness on substantially larger or smaller models remains to be validated. Following the scaling findings of SDPO (Hübotter et al., 2026), we expect the benefit to grow with model scale, as stronger in-context learners produce more reliable self-teacher signals, but this hypothesis requires empirical confirmation.

Cross-model ablation coverage. Our full ablation suite is conducted only on Qwen3-8B. OLMo-3-7B is limited to the core SDPO vs. SC-SDPO comparison, which suffices to confirm cross-family generalization of the proposed weighting but leaves open whether the relative ranking among baselines transfers across architectures. Extending the complete ablation to additional model families is a natural direction for future work.

References

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. 2024. On-policy distillation of language models: Learning from self-generated mistakes. In *International Conference on Learning Representations*, volume 2024, pages 21246–21263.

Sanghwan Bae, Jiwoo Hong, Min Young Lee, Hanbyul Kim, JeongYeon Nam, and Donghyun Kwak. 2026. Online difficulty filtering for reasoning oriented reinforcement learning. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 700–719.

Yanqi Dai, Yuxiang Ji, Xiao Zhang, Yong Wang, Xiangxiang Chu, and Zhiwu Lu. 2026. Harder is better: Boosting mathematical reasoning via difficulty-aware grpo and multi-aspect question reformulation. *arXiv preprint arXiv:2601.20614*.

DeepSeek-AI. 2026. Deepseek-v4: Towards highly efficient million-token context intelligence.

Ken Ding. 2026. Hdpo: Hybrid distillation policy optimization via privileged self-distillation. *arXiv preprint arXiv:2603.23871*.

Kehua Feng, Xinyi Shen, Weijie Wang, Xiang Zhuang, Yuqi Tang, Qiang Zhang, and Keyan Ding. 2024. Sciknoweval: Evaluating multi-level scientific knowledge of large language models. *arXiv preprint arXiv:2406.09098*.

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

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, and 1 others. 2025. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638.

Yinghui He, Simran Kaur, Adithya Bhaskar, Yongjin Yang, Jiarui Liu, Narutatsu Ri, Liam Fowl, Abhishek Panigrahi, Danqi Chen, and Sanjeev Arora. 2026. Self-distillation zero: Self-revision turns binary rewards into dense supervision. *arXiv preprint arXiv:2604.12002*.

Jonas Hübotter, Frederike Lübeck, Lejs Behric, Anton Baumann, Marco Bagatella, Daniel Marta, Ido Hakimi, Idan Shenfeld, Thomas Kleine Buening, Carlos Guestrin, and 1 others. 2026. Reinforcement learning via self-distillation. *arXiv preprint arXiv:2601.20802*.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, and 1 others. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.

Gengsheng Li, Tianyu Yang, Junfeng Fang, Mingyang Song, Mao Zheng, Haiyun Guo, Dan Zhang, Jinqiao Wang, and Tat-Seng Chua. 2026. Unifying group-relative and self-distillation policy optimization via sample routing. *arXiv preprint arXiv:2604.02288*.

Kevin Lu and Thinking Machines Lab. 2025. On-policy distillation. *Thinking Machines Lab: Connectionism*. Https://thinkingmachines.ai/blog/on-policy-distillation.

Team Olmo, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, and 1 others. 2025. Olmo 3. *arXiv preprint arXiv:2512.13961*.

Hejian Sang, Yuanda Xu, Zhengze Zhou, Ran He, Zhipeng Wang, and Jiachen Sun. 2026. On-policy self-distillation for reasoning compression. *arXiv preprint arXiv:2603.05433*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

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

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

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

Qiaoyu Tang, Ziliang Deng, Hongyu Lin, Xianpei Han, Qiao Liang, Boxi Cao, and Le Sun. 2023. Toolalpaca: Generalized tool learning for language models with 3000 simulated cases. *arXiv preprint arXiv:2306.05301*.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, and 1 others. 2025. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*.

Jiaxuan Wang, Xuan Ouyang, Zhiyu Chen, Yulan Hu, Zheng Pan, Xin Li, and Lan-Zhe Guo. 2026. Trace: Distilling where it matters via token-routed self on-policy alignment. *arXiv preprint arXiv:2605.10194*.

Bangjun Xiao, Bingquan Xia, Bo Yang, Bofei Gao, Bowen Shen, Chen Zhang, Chenhong He, Chiheng Lou, Fuli Luo, Gang Wang, and 1 others. 2026. Mimo-v2-flash technical report. *arXiv preprint arXiv:2601.02780*.

Yuanda Xu, Hejian Sang, Zhengze Zhou, Ran He, and Zhipeng Wang. 2026. PACED: Distillation and on-policy self-distillation at the frontier of student competence. *arXiv preprint arXiv:2603.11178*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Chenxu Yang, Chuanyu Qin, Qingyi Si, Minghui Chen, Naibin Gu, Dingyu Yao, Zheng Lin, Weiping Wang, Jiaqi Wang, and Nan Duan. 2026a. Self-distilled RLVR. *arXiv preprint arXiv:2604.03128*.

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

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, and 1 others. 2026. Dapo: An open-source llm reinforcement learning system at scale. *Advances in Neural Information Processing Systems*, 38:113222–113244.

Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, and 1 others. 2026. Glm-5: from vibe coding to agentic engineering. *arXiv preprint arXiv:2602.15763*.

Jixiao Zhang and Chunsheng Zuo. 2025. Grpo-lead: A difficulty-aware reinforcement learning approach for concise mathematical reasoning in language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 5642–5665.

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

Binbin Zheng, Xing Ma, Yiheng Liang, Jingqing Ruan, Xiaoliang Fu, Kepeng Lin, Benchang Zhu, Ke Zeng, and Xunliang Cai. 2026. Scope: Signal-calibrated on-policy distillation enhancement with dual-path adaptive weighting. *arXiv preprint arXiv:2604.10688*.

A Appendix

A.1 Dataset Details

We use the same train/test splits provided by Hübotter et al. (2026), summarized in Table 3.

Dataset	Source	Train	Test	Total
Chemistry	SciKnowEval	1,890	210	2,100
Physics	SciKnowEval	720	80	800
Biology	SciKnowEval	450	50	500
Materials	SciKnowEval	841	94	935
Tool Use	ToolAlpaca	4,046	68	4,114

Table 3: Dataset statistics.

A.2 Hyperparameters

We summarize hyperparameters in table 4.

Parameters	
General	
Model	Qwen/Qwen3-8B
	allenai/Olmo3-7B-Instruct
Thinking	False
Data	
Max. prompt length	2048
Max. response length	2048
	8192 (GRPO)
Batching	
Question batch size	32
Mini batch size	32
Number of rollouts	8
Rollout	
Inference engine	vllm
Temperature	1.0
Validation	
Number of rollouts	16
Temperature	0.6
Top-p	0.95
Loss	
Top-K distillation	100
Distillation divergence	Jensen–Shannon
Teacher-EMA update rate	0.05
ϵ-high	0.28 (GRPO)
KL coefficient (λ)	0 (GRPO)
Normalize advantages	True (GRPO)
	False (GRPO w/o Norm)
Training	
Optimizer	AdamW
Learning rate	1×10^{-5} (constant)
Warmup steps	10
Weight decay	0.01
Gradient Clip Norm	1.0

Table 4: Hyperparameters.

A.3 Per-Token Advantage under Jensen–Shannon Divergence

Let $\pi_S := \pi_\theta(\cdot \mid x, y_{i,<t})$ and $\pi_T := \pi_\theta(\cdot \mid x, f_i, y_{i,<t})$ denote the student and teacher distributions, and $M := \frac{1}{2}(\pi_S + \pi_T)$ their mixture. The Jensen–Shannon divergence is

$$D_{\mathrm{JSD}}(\pi_S \| \pi_T) = \frac{1}{2} D_{\mathrm{KL}}(\pi_S \| M) + \frac{1}{2} D_{\mathrm{KL}}(\pi_T \| M). \tag{17}$$

Taking the gradient with respect to θ (with stop-gradient on π_T), the mixture $M = \frac{1}{2}(\pi_S + \pi_T)$ depends on π_S, so both KL terms contribute.

To derive the gradient, we first expand the divergence as a sum over the vocabulary y (omitting the conditioning variables for brevity):

$$D_{\mathrm{JSD}} = \sum_y \left[\frac{1}{2} \pi_S(y) \log \frac{\pi_S(y)}{M(y)} + \frac{1}{2} \pi_T(y) \log \frac{\pi_T(y)}{M(y)} \right]. \tag{18}$$

We compute the partial derivative of the divergence with respect to the student probability $p = \pi_S(y)$. Let $q = \pi_T(y)$, and the mixture distribution be $m = M(y) = \frac{p+q}{2}$. Applying the product and chain rules, and noting that $\frac{\partial m}{\partial p} = \frac{1}{2}$, we differentiate the terms inside the summation:

$$\frac{\partial}{\partial p} \left(\frac{1}{2} p \log \frac{p}{m} + \frac{1}{2} q \log \frac{q}{m} \right). \tag{19}$$

Evaluating the derivative term by term yields:

1. $\frac{\partial}{\partial p} \left(\frac{1}{2} p \log p \right) = \frac{1}{2} (\log p + 1)$
2. $\frac{\partial}{\partial p} \left(-\frac{1}{2} p \log m \right) = -\frac{1}{2} \log m - \frac{p}{4m}$
3. $\frac{\partial}{\partial p} \left(\frac{1}{2} q \log q \right) = 0$
4. $\frac{\partial}{\partial p} \left(-\frac{1}{2} q \log m \right) = -\frac{q}{4m}$

Summing these partial derivatives results in:

$$\begin{aligned} \frac{\partial D_{\mathrm{JSD}}}{\partial p} &= \frac{1}{2} \log \frac{p}{m} + \frac{1}{2} - \frac{p+q}{4m} \\ &= \frac{1}{2} \log \frac{p}{m} + \frac{1}{2} - \frac{2(p+q)}{4(p+q)} \\ &= \frac{1}{2} \log \frac{p}{m} + \frac{1}{2} - \frac{1}{2} \\ &= \frac{1}{2} \log \frac{p}{m}. \end{aligned} \tag{20}$$

Thus:

$$\begin{aligned} \nabla_\theta D_{\mathrm{JSD}} &= \sum_y \frac{\partial D_{\mathrm{JSD}}}{\partial \pi_S(y)} \nabla_\theta \pi_S(y) \\ &= \sum_y \frac{1}{2} \log \frac{\pi_S(y)}{M(y)} \nabla_\theta \pi_S(y) \end{aligned} \tag{21}$$

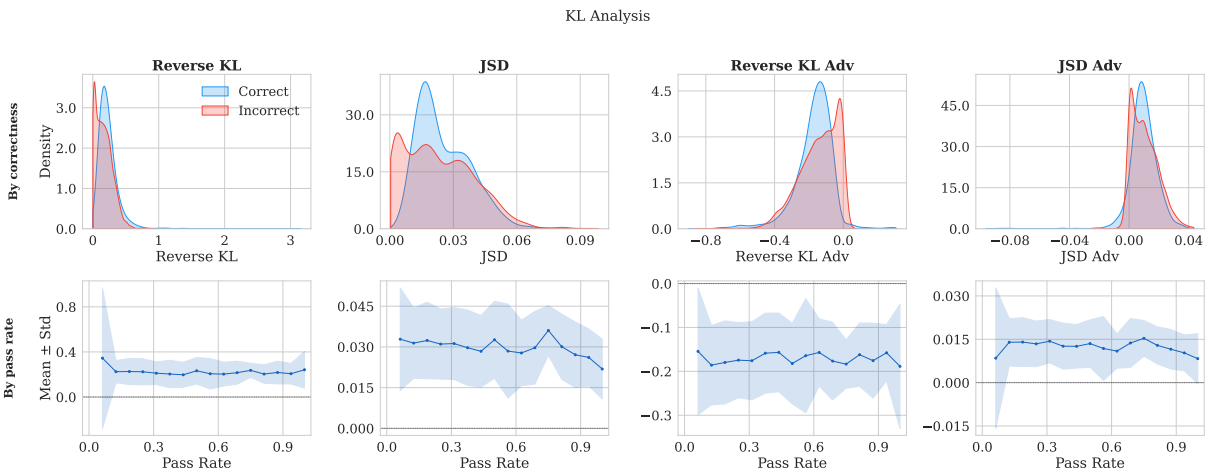


Figure 4: KL-based analysis of the base model (`Qwen3-8B`). **Top row**: kernel density estimates of each metric, stratified by whether the individual rollout is correct (blue) or incorrect (red). **Bottom row**: per-metric mean $\pm$ one standard deviation as a function of the question-level pass rate (fraction of correct rollouts among 16, ranging from 0.0625 to 1).

Using the log-derivative trick, $\nabla_\theta \pi_S(y) = \pi_S(y) \nabla_\theta \log \pi_S(y)$, we can rewrite the gradient as:

$$\nabla_\theta D_{\text{JSD}} = \sum_y \pi_S(y) \left[\frac{1}{2} \log \frac{\pi_S(y)}{M(y)} \right] \nabla_\theta \log \pi_S(y)$$
$$= \mathbb{E}_{y \sim \pi_S} \left[\frac{1}{2} \log \frac{\pi_S(y)}{M(y)} \nabla_\theta \log \pi_S(y) \right]. \tag{22}$$

The per-token advantage ($A(y)$) is given as:

$$A_{\text{JSD}}(y) = \frac{1}{2} \log \frac{\pi_S(y)}{M(y)}. \tag{23}$$

A.4 KL Analysis

To empirically verify that SDPO's distillation loss does not implicitly recover question-level signal modulation, we analyze the distribution of several KL-based metrics as a function of pass rate on the base model (Qwen3-8B).

We sample n=16 rollouts per question (temperature 0.6, top-p 0.95). The teacher distribution is obtained via the SDPO reprompt template. All distributional metrics are computed over the student's top-100 vocabulary indices with an additional tail bucket. We consider four metrics: the reverse KL divergence $\text{KL}(\pi_\theta \| \pi_t)$, the Jensen–Shannon divergence $\text{JSD} = \frac{1}{2}\text{KL}(\pi_\theta \| M) + \frac{1}{2}\text{KL}(\pi_t \| M)$ with $M = \frac{1}{2}(\pi_\theta + \pi_t)$, and their token-level advantage counterparts: $\log \frac{\pi_t(\hat{y}_t)}{\pi_\theta(\hat{y}_t)}$ and $\frac{1}{2} \log \frac{\pi_\theta(\hat{y}_t)}{M(\hat{y}_t)}$, each averaged over response tokens.

As shown in Figure 4 (bottom row), all four metrics remain approximately flat as a function of pass rate. Note that questions with $\hat{p} = 0$ are excluded from this analysis, since the self-teacher requires a correct rollout as the reference demonstration and no such rollout exists when all attempts fail.

A.5 Learnability Bound under Normalized Rewards

The reverse KL bound of Bae et al. (2026) (Eq. 6) is derived under the raw binary reward $r \in \{0, 1\}$. Here we extend their CGF framework (Proposition 6.1 in Bae et al. 2026) to the standardized reward that GRPO's group-relative normalization implicitly operates on.

Proposition 1. *Let $r \in \{0, 1\}$ with $\mathbb{E}[r] = p \in (0, 1)$. Define $\tilde{r} := (r - p)/\sqrt{p(1-p)}$ and the corresponding optimal policy $\tilde{\pi}^* \propto \pi_{\text{init}} \exp(\tilde{r}/\beta)$. Then*

$$D_{\text{KL}}(\pi_{\text{init}} \,\|\, \tilde{\pi}^*) = \frac{1}{2\beta^2} + O(\beta^{-3}). \tag{24}$$

Proof. By the CGF identity of Bae et al. (2026), $D_{\text{KL}}(\pi_{\text{init}} \| \tilde{\pi}^*) = K_{\tilde{r}}(1/\beta)$, where $K_Z(t) = \sum_{k=2}^{\infty} \kappa_k t^k / k!$ for a zero-mean variable Z. The standardized reward takes values $\sqrt{(1-p)/p}$ with probability p and $-\sqrt{p/(1-p)}$ with probabil-

ity $1-p$, giving:

$$
\begin{aligned}
K_{\tilde{r}}\left(\tfrac{1}{\beta}\right) &= \frac{\kappa_2}{2\beta^2} + O(\beta^{-3}) \\
&= \frac{\mathrm{Var}[\tilde{r}]}{2\beta^2} + O(\beta^{-3}) \\
&= \frac{p \cdot \frac{1-p}{p} + (1-p) \cdot \frac{p}{1-p}}{2\beta^2} + O(\beta^{-3}) \\
&= \frac{1}{2\beta^2} + O(\beta^{-3}).
\end{aligned}
\tag{25}
$$

□

Comparing with the raw-reward bound $D_{\mathrm{KL}} \geq p(1-p)/(2\beta^2)$ of Bae et al. (2026), the leading term under normalized rewards becomes $1/(2\beta^2)$—a constant independent of p. This reveals that GRPO's group-relative normalization acts as an implicit difficulty-aware mechanism: by dividing the reward by $\sqrt{p(1-p)}$, it equalizes the leading-order learnability across all non-degenerate questions ($0 < p < 1$), absorbing the $p(1-p)$ factor that appeared in the raw-reward bound. Independently, the per-step gradient magnitude is exactly $\sqrt{p(1-p)}$ by the algebraic identity in Eq. 5, supporting $\alpha = 1/2$ as the scale-consistent exponent.

A.6 Pass Rate Analysis

Figure 5a tracks the per-batch pass rate distribution over 200 training steps. On most datasets, the model's growing competence gradually converts intractable questions ($\hat{p} = 0$, red) into solvable ones, expanding the green (mastered) region over time. On Biology, SC-SDPO ($\alpha = 0.5$) maintains a visibly broader band of intermediate pass rates ($\hat{p} \in [1/8, 7/8]$) compared to the SDPO baseline, indicating a larger effective learnable zone throughout training. ToolUse presents a distinct pattern: pass rates for all methods remain heavily concentrated at the extremes ($\hat{p} \approx 0$ or $\hat{p} \approx 1$), leaving a narrow intermediate region and limited room for difficulty-aware weighting to differentiate methods.

Figure 5b quantifies the fraction of training samples in the intermediate pass-rate bins across 50-step windows. SRPO maintains the highest intermediate fraction throughout training in both the broader ($\hat{p} \in [1/8, 7/8]$) and narrower ($\hat{p} \in [2/8, 6/8]$) ranges. This is expected given SRPO's routing mechanism: by directing incorrect rollouts with correct siblings to the SDPO branch and all others to GRPO, it effectively preserves a large pool of mixed-outcome groups rather than resolving them toward mastery. However, as shown in the main results, this higher intermediate retention does not fully translate into superior accuracy, suggesting that sustaining a large sweet spot is necessary but not sufficient. Among non-routing methods, SC-SDPO ($\alpha = 0.5$) achieves the highest intermediate fraction in the early phase of training (steps 1–100), where the learning frontier is broadest and difficulty-aware weighting has the greatest impact. As training progresses and more questions are mastered, the intermediate fraction naturally declines across all methods; SC-SDPO converges smoothly, consistent with the implicit curriculum effect of our batch-adaptive design.

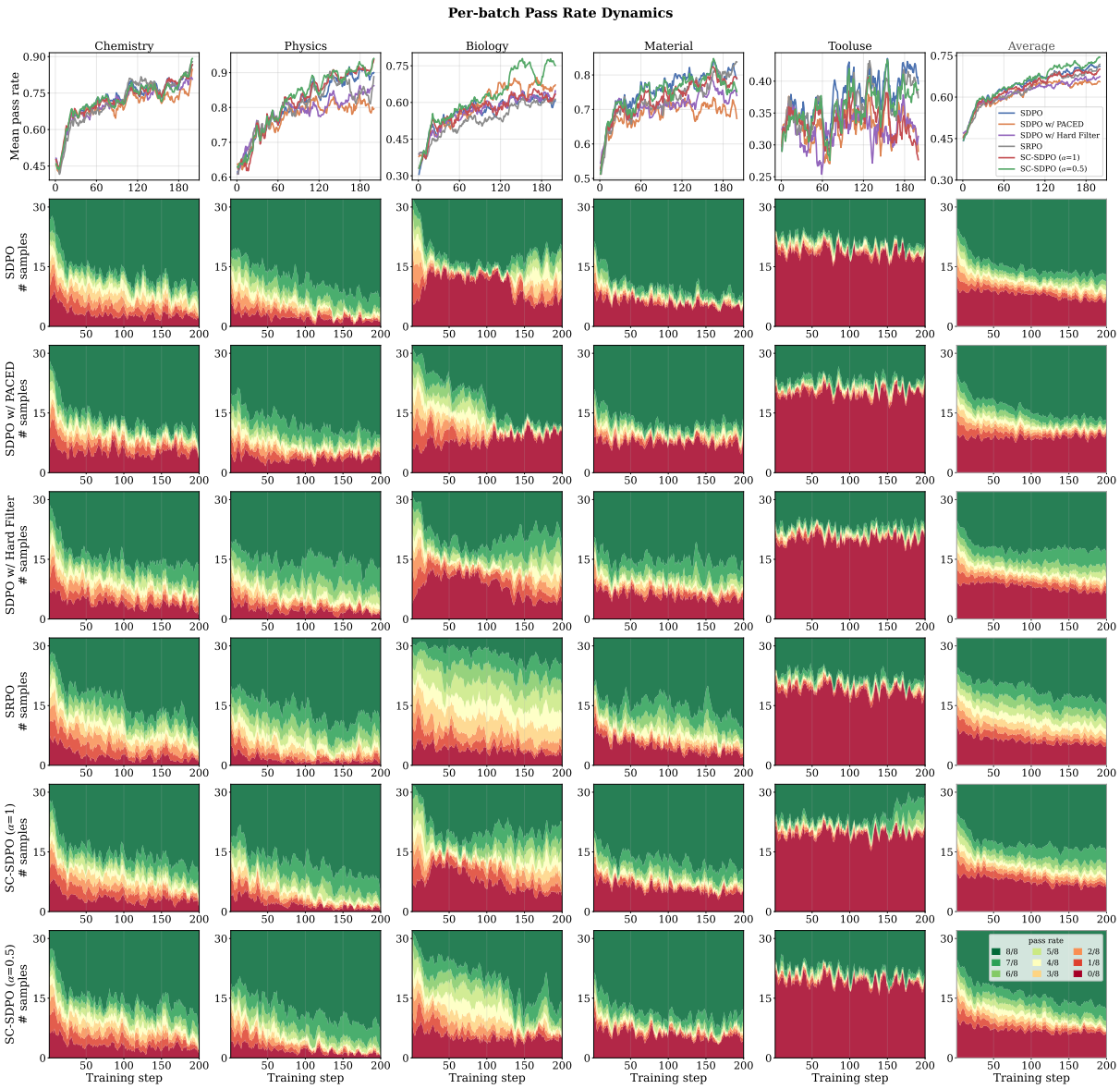
A.7 Prompt Templates

We adopt the same prompt templates as Hübotter et al. (2026).

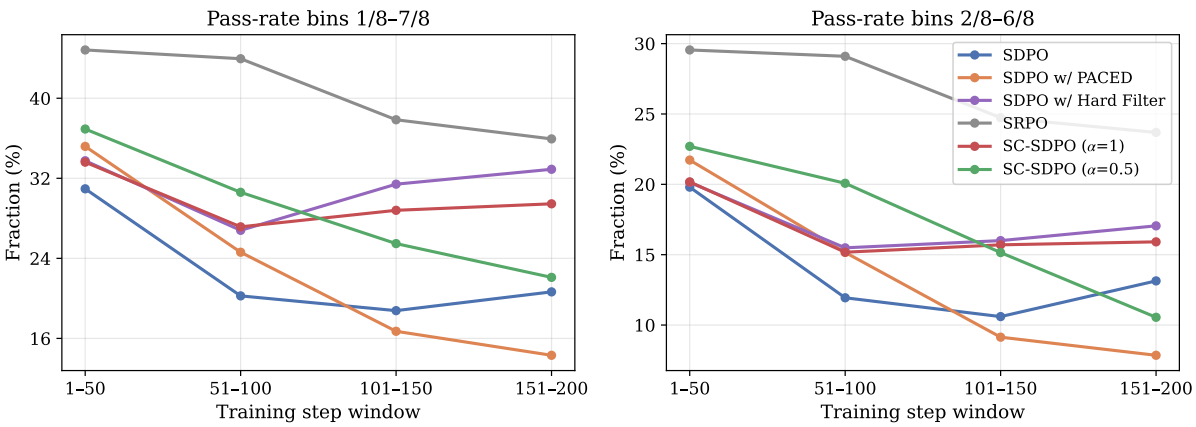
Science QA Prompt (SciKnowEval) Science QA tasks use a fixed system prompt combined with a per-question user message.

System Prompt

```
Given a question and four options,
please select the right answer.
Respond in the following format:
<reasoning>
...
</reasoning>
<answer>
...
</answer>
For the answer, only output the
letter corresponding to the correct
option (A, B, C, or D), and nothing
else. Do not restate the answer
text. For example, if the answer is
"A", just output:
<answer>
A
</answer>
```



(a) Per-batch pass rate dynamics over 200 training steps across five datasets and their average (rightmost column). **Row 1:** Mean pass rate per training batch for each method, smoothed with a sliding window. **Rows 2–7:** Stacked area charts showing the sample count distribution across nine discrete pass rate levels $(0, \frac{1}{8}, \ldots, 1)$ at each step, color-coded from red (pass rate $= 0$, all rollouts incorrect) to green (pass rate $= 1$, all rollouts correct), smoothed with a sliding window.



(b) Fraction of training samples in pass-rate bins 1/8–7/8 (left) and bins 2/8–6/8 (right) per 50-step training window, averaged across five datasets.

User Message Template

```
{question}
A: {choice_A}
B: {choice_B}
C: {choice_C}
D: {choice_D}
Please reason step by step.
```

Teacher Context — With Solution

[User]

```
{problem}
Please reason step by step, and put
your final answer within \boxed{}.
Correct solution:
{successful_previous_attempt}
Correctly solve the original
question.
```

[Assistant]

{student_response $\hat{y}$}

Tool Use Prompt Tool use tasks have no separate system prompt. The entire context is contained in a single user message structured as follows:

Tool Use Prompt Template

```
Your task is to answer the user's
question using available tools.
You have access to the following
tools:
Name: {tool_name}
Description: {tool_description}
Documentation:
{method_name}:
{method_description}
Parameters: {parameters_json}
Output: {output_description}
[... additional tools ...]

Use the following format:
Thought: you should always think
about what to do
Action: the action to take, should
be one of the tool names.
Action Input: the input to the
action, must be in JSON format.
All of the action input must be
realistic and from the user.

Begin!
Question: {user_question}
```

The expected response follows the `Thought / Action / Action Input` format. The verifier checks two conditions jointly: (1) the predicted action names match the ground-truth action names (multiset equality), and (2) the merged JSON parameters across all `Action Input` blocks exactly match the ground-truth parameters. Both conditions must hold for a reward of 1 (correct); otherwise the reward is 0 (incorrect).

When SDPO self-distillation is applied to tool use, the feedback section in the teacher context is populated with a natural-language description of the mismatch, e.g.:

Feedback Example (Tool Use)

```
Action inputs mismatch: predicted
{"method": "GET", ...},
expected {"method": "POST", ...}
```