SUCCESS-AWARE POLICY LEARNING IN RLVR

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning with verifiable rewards (RLVR) remains sample-inefficient on hard prompts, where most rollouts fail and the learner receives only a binary correctness signal. In the standard sample–verify–update loop, a rare successful rollout is used primarily as a target for parameter updates, leaving its reasoning content unused for acquiring subsequent training experience. We introduce *Success-Aware Policy Learning* (SAPL), which gives each verified success a second role of temporary in-context guidance for further sampling before the policy is updated. SAPL first discovers verifier-correct on-policy rollouts and selects the success with the highest likelihood under the current policy. The same model, with unchanged parameters, then conditions on this success to generate additional candidates. The verifier filters these candidates, and a weighted objective assimilates the verified responses into the prompt-only policy. SAPL requires no reference solution, external teacher, or verifier feedback beyond correctness, and uses no auxiliary context at inference. We show that SAPL achieves improved accuracy across Qwen3-4B, 8B, and 14B on eleven competition mathematics benchmarks by 4.6–8.7 percentage points over the strongest evaluated RLVR baseline and by 2.6–5.6 points over the strongest evaluated ground-truth-conditioned baseline. These results show that rare successes are valuable not only as update targets, but also as guides for acquiring new verified experience.

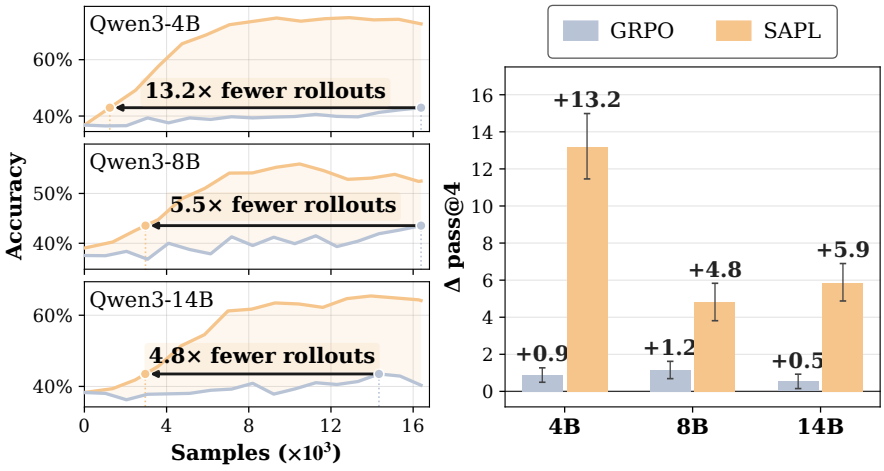

Figure 1: **SAPL converts rare on-policy successes into more sample-efficient learning. Left:** Validation accuracy versus the cumulative number of responses generated during training on hard prompts. Under a matched sampling budget, SAPL improves rapidly across all three model scales, whereas GRPO remains close to the base model. The annotations report the reduction in rollouts achieved by SAPL at the matched accuracy levels used for comparison. **Right:** Improvement in pass@4 over the base model, averaged over MATH-500 and OlympiadBench. SAPL provides larger gains than GRPO at every model scale, with an average advantage of 7.1 percentage points.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

1 INTRODUCTION

Reinforcement learning from verifier rewards (RLVR) has become a powerful approach for improving language model reasoning on tasks with automatically verifiable outcomes, such as mathematics and code (Shao et al., 2024; Guo et al., 2025). However, RLVR remains highly sample inefficient on difficult problems. When the policy has low pass@1, most sampled responses fail, most rollout groups contain no positive task signal, and the few informative groups typically contain only a single successful response. This challenge is especially severe in the verifier-only setting, where the learner receives no reference solution, reasoning critique, or dense feedback, but only receives a scalar correctness signal.

The existing approaches (see Appendix A.1 for detailed context) address this bottleneck along several axes. Policy-gradient methods modify advantage estimation or increase the influence of rare successes; replay methods reuse previously successful responses; and recent success-guided or self-distillation methods use successful trajectories as hints or feedback for learning (Tajwar et al., 2026; Zhan et al., 2026; Qiyuan et al., 2026; Hübotter et al., 2026; Yu et al., 2026b; Chakraborty et al., 2026). These directions show that successful responses can be reused, but they also expose two distinct roles that a response can play. Its *optimization value* comes from identifying behavior whose likelihood should be increased. Its *data-acquisition value* comes from being a complete, model-generated solution that can guide where the model samples next. We study a specific post-success opportunity in which, after a complete on-policy success has been found, can it be used to generate additional verifier-accepted responses before the policy is updated, and can those responses be learned without requiring the successful response at inference time?

Our key insight. A useful perspective comes from the binary-reward policy gradient. For the baseline-free estimator studied in Section 2, sampling and verification are followed by a supervised log-likelihood update on the observed successes, scaled by the fraction of responses that succeeded. This connects RLVR to learning from success-conditioned responses (Russo, 2026). The complete successful response already serves as an update target, but in the conventional sample–verify–update loop, its reasoning content affects subsequent sampling only after it has been compressed into a parameter change. But for language models through in-context learning (Brown et al., 2020), the same response can immediately change the distribution of subsequent generations without changing the model parameters. Our key insight is to use a verified response not only as a target for learning, but also as temporary context for acquiring further successful experience before the parameter update.

We introduce *Success-Aware Policy Learning* (SAPL), an iterative *discover–amplify–assimilate* procedure. During *discovery*, the current policy generates on-policy responses, and the verifier identifies the successful ones. When several successes are available, SAPL selects the response with the highest likelihood under the unguided policy, favoring a solution that the model already finds plausible. During *amplification*, the same model, with unchanged parameters, receives this response as temporary context and generates fresh full-response candidates. Finally, during *assimilation*, the guide is removed and the verified guided responses are learned from using the original prompt alone. A weighted objective down-weights tokens that remain highly unlikely under the unguided policy, reducing the mismatch between guided generation and prompt-only inference (Chakraborty et al., 2026). We summarize our contributions as follows:

- **Success-aware data acquisition.** We distinguish the optimization value of a verified success from its data-acquisition value, and study how the latter can improve RLVR under a verifier-only interface. The proposed mechanism uses success to influence additional sampling immediately, before a parameter update.
- **Success-aware policy learning.** We introduce SAPL, which combines online discovery and likelihood-based selection of an on-policy success, full-response generation conditioned on that success, independent verification, and prompt-only surprisal-gated assimilation. The procedure uses a single model and preserves standard inference.
- **Evaluation across model scales.** Experiments with Qwen3-4B, 8B, and 14B show improvements in held-out validation accuracy (Figure 1) and downstream reasoning performance. Across eleven competition mathematics benchmarks, SAPL exceeds the strongest evaluated RLVR baseline at each scale by 4.6–8.7 percentage points in average pass@8. It also exceeds the strongest evaluated answer-conditioned baseline by 2.6–5.6 points, while using only verifier feedback as external supervision.

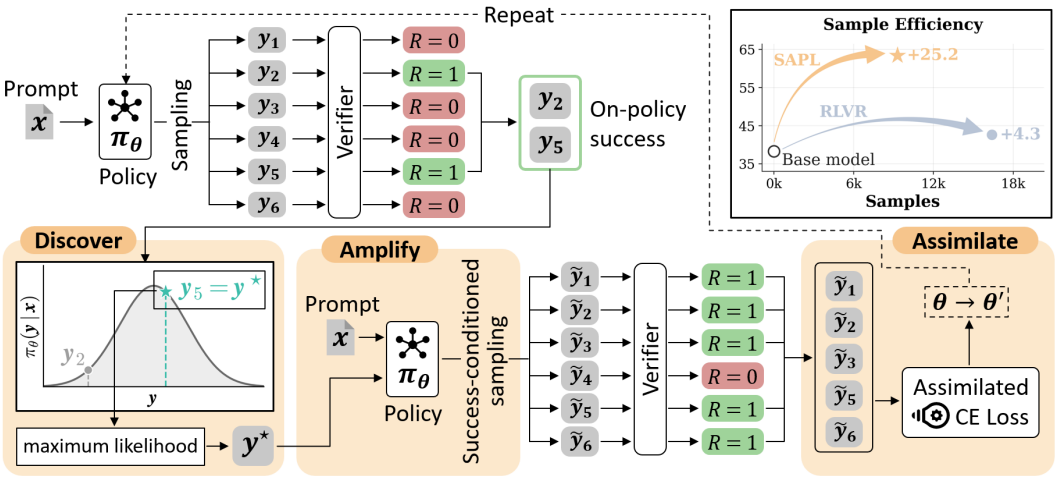

Figure 2: **SAPL adds a direct path from verified success to data acquisition. Top:** In the conventional sample–verify–update loop, the policy generates on-policy responses, the verifier returns binary rewards, and successful responses affect later sampling through a parameter update. On hard prompts, most rollout groups contain no success, leaving little task-relevant learning signal. **Bottom:** SAPL first *discovers* verifier-correct responses and selects the one with the highest length-normalized likelihood. It then *amplifies* this success by using it as temporary context for generating fresh responses from the same unchanged model, each of which is independently verified. Finally, SAPL removes the guide and *assimilates* the verified guided responses into the prompt-only policy. The cycle repeats without external solutions or auxiliary context at inference.

2 Problem Formulation

Verifier-only RLVR. We study reinforcement learning from verifier rewards (RLVR) for post-training language models on reasoning tasks with verifiable outcomes. Given a prompt $\mathbf{x} \sim \mathcal{D}$, an LLM policy $\pi_\theta(\cdot|\mathbf{x})$ generates a complete response $\mathbf{y} = (y_1, \ldots, y_T)$ autoregressively, $y_t \sim \pi_\theta(\cdot|\mathbf{x}, \mathbf{y}_{<t})$. A task-specific verifier returns a binary reward $R(\mathbf{x}, \mathbf{y}) \in \{0, 1\}$ indicating whether the response is correct. For example, in math tasks, the verifier extracts the final answer from the response and compares it with the ground truth. The reward, therefore, depends only on the final answer, not on the reasoning that leads to it. We consider a *verifier-only* interface in which we observe only scalar reward and no other feedback is obtained. The success probability of the policy on a prompt (its pass@1) and the post-training objective $J_{\mathrm{RLVR}}(\theta)$ are

$$p_\theta(\mathbf{x}) := \mathbb{E}_{\mathbf{y}\sim\pi_\theta(\cdot|\mathbf{x})}\big[R(\mathbf{x}, \mathbf{y})\big], \quad \text{and} \quad J_{\mathrm{RLVR}}(\theta) := \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\big[p_\theta(\mathbf{x})\big]. \tag{1}$$

Learning from verified successes. RLVR improves the objective in equation (1) by gradient ascent. To see how verified responses enter the policy gradient $\nabla_\theta J_{\mathrm{RLVR}} = \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}[\nabla_\theta p_\theta(\mathbf{x})]$, applying the log-derivative trick gives

$$\nabla_\theta p_\theta(\mathbf{x}) = \mathbb{E}_{\mathbf{y}\sim\pi_\theta(\cdot|\mathbf{x})}\big[R(\mathbf{x}, \mathbf{y})\, \nabla_\theta \log \pi_\theta(\mathbf{y}|\mathbf{x})\big]. \tag{2}$$

In practice, a baseline-free Monte Carlo estimator draws K independent on-policy responses $\mathbf{y}_1, \ldots, \mathbf{y}_K \sim \pi_\theta(\cdot|\mathbf{x})$ and averages their reward-weighted log-likelihood gradients:

$$\hat{g}_K(\mathbf{x}) = \frac{1}{K} \sum_{i=1}^{K} R(\mathbf{x}, \mathbf{y}_i)\, \nabla_\theta \log \pi_\theta(\mathbf{y}_i|\mathbf{x}). \tag{3}$$

Because the reward is binary, every unsuccessful response is multiplied by zero in equation (3), and only successful responses remain. To make this explicit, let $\mathcal{S}_K$ be the set of successful rollouts in the group and $N_K = |\mathcal{S}_K|$ its size. Whenever $N_K > 0$, equation (3) can be rewritten as

$$\hat{g}_K(\mathbf{x}) = \frac{N_K}{K} \left[\frac{1}{N_K} \sum_{i\in\mathcal{S}_K} \nabla_\theta \log \pi_\theta(\mathbf{y}_i|\mathbf{x}) \right]. \tag{4}$$

For a group of K samples, the term in brackets is the gradient of the mean sequence log-likelihood of its successful responses. Thus, the baseline-free policy-gradient update using equation (4) is equivalent to a supervised log-likelihood update on the observed successes, scaled by the fraction of responses that succeeded. Although this identity is specific to the baseline-free RLVR estimator, it exposes a useful connection between *sampling* and *learning* as the policy must first generate verified successes before they can contribute to this update.

The low-pass@1 bottleneck and sample inefficiency. This dependence becomes restrictive on hard prompts, where $p_\theta(\mathbf{x})$ is small. Under independent sampling at a fixed policy, $N_K \sim \mathrm{Binomial}(K, p_\theta(\mathbf{x}))$, so $\mathbb{E}[N_K] = Kp_\theta(\mathbf{x})$ and $\Pr(N_K = 0) = (1 - p_\theta(\mathbf{x}))^K$. In the regime $Kp_\theta(\mathbf{x}) \ll 1$, most sampled groups contain no successful response and yield a zero gradient estimate in equation (4), while nonzero estimates typically rely on a single successful response. The learning signal is therefore infrequent and supported by limited successful experience. We further note that obtaining more successes is costly because for $p_\theta(\mathbf{x}) > 0$, finding even one verified success requires $1/p_\theta(\mathbf{x})$ rollouts in expectation. Hence, we ask: can we improve the sample efficiency of RLVR on hard prompts, achieving greater gains in pass@1 under limited rollout budget while retaining scalar verifier feedback as the only external supervision.

3 Proposed Approach: Success-Aware Policy Learning

Section 2 separates RLVR into two operations: acquiring verifier-accepted responses and updating the policy from them. In the low-pass@1 regime, the first operation is costly because a limited rollout budget yields few successes from which to learn. This success-imitation view of the update connects to the broader analysis of success conditioning by Russo (2026).

Our key insight: Although the full correct response in equation (4) contributes to this parameter update, its content is not used directly to guide additional sampling before the parameters change (cf. Figure 2). This distinction is important in our setting because the mathematics verifier checks the final answer, but a response that passes this check also contains the model-generated reasoning that led to it. *Our key insight is to use this response not only as a learning target, but also as temporary guidance for acquiring further successful experience.*

Based on this insight, we propose success-aware policy learning (SAPL) which introduces a direct post-success path from verification back to sampling. Once the current policy discovers a verified response, SAPL uses that response as temporary context for the same, unchanged model, allowing it to influence subsequent sampling before any parameter update. The resulting candidates are independently verified and then assimilated into the prompt-only policy. SAPL proceeds through three stages: *discover*, *amplify*, and *assimilate*.

(1) Discover: obtain and select an on-policy success. As in RLVR, the policy samples K responses for each prompt $\mathbf{x}$, and the verifier checks them. Let $\mathcal{S}_K(\mathbf{x})$ be the set of successful responses. On hard prompts, this set is often empty. When it is not, it usually contains only a few responses. Prompts with no success have no demonstration to offer, so SAPL skips them in this iteration.

When multiple successes are observed, SAPL selects the response that the current prompt-only policy finds most likely on average per token:

$$\mathbf{y}^\star = \arg\max_{\mathbf{y} \in \mathcal{S}_K(\mathbf{x})} \frac{1}{|\mathbf{y}|} \log \pi_\theta(\mathbf{y}|\mathbf{x}). \tag{5}$$

The length normalization reduces the systematic preference for shorter responses induced by total sequence log-likelihood. As illustrated in Figure 3, a verified response that is more likely under the current policy also provides more effective in-context guidance on the

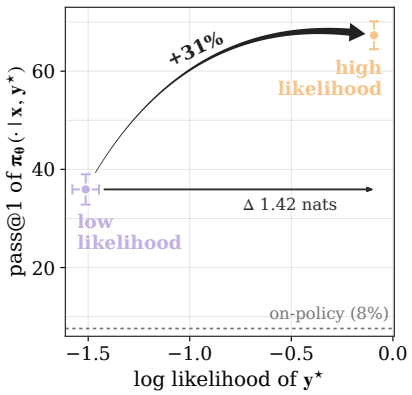

Figure 3: Higher-likelihood verified responses provide stronger guidance: On the analyzed subset of hard mathematics prompts, the unguided policy has 8% pass@1. Conditioning the unchanged model on a lower-likelihood verified response yields 36% guided pass@1, whereas a higher-likelihood response yields 67%. This motivates the high likelihood success in equation (5).

Algorithm 1 Success-aware Policy Learning (SAPL)

Require: Pretrained LLM π_θ; hard dataset $\mathcal{D}$; verifier $r(\cdot) \in \{0, 1\}$; prompts batch size B; group size K; success-aware group size M; gate sharpness κ; midpoint γ; inner steps μ; iterations N

for $n \leftarrow 1$ **to** N **do**
  Sample a batch of prompts $\{\mathbf{x}_b\}_{b=1}^{B} \sim \mathcal{D}$; $\quad \mathcal{B}_n \leftarrow \emptyset$
  Sample on-policy solutions:
  for $b \leftarrow 1$ **to** B **do**
    $\{\mathbf{y}_i\}_{i=1}^{K} \sim \pi_\theta(\cdot|\mathbf{x}_b)$
    $\mathcal{S}_b \leftarrow \{\mathbf{y}_i : r(\mathbf{x}_b, \mathbf{y}_i) = 1\}$; $\quad \mathbf{y}_b^\star \leftarrow \arg\max_{\mathbf{y} \in \mathcal{S}_b} \frac{1}{|\mathbf{y}|} \log \pi_\theta(\mathbf{y}|\mathbf{x}_b)$
  end for
  Sample success-aware solutions:
  for each $\mathbf{x}_b$ with $\mathcal{S}_b \neq \emptyset$ **do**
    $\{\tilde{\mathbf{y}}_j\}_{j=1}^{M} \sim \pi_\theta(\cdot|\mathbf{x}_b, \mathbf{y}_b^\star)$
    $\mathcal{C}_b \leftarrow \{\tilde{\mathbf{y}}_j : r(\mathbf{x}_b, \tilde{\mathbf{y}}_j) = 1\}$
    $\mathcal{B}_n \leftarrow \mathcal{B}_n \cup \big(\{\mathbf{x}_b\} \times \mathcal{C}_b\big)$
  end for
  Parameter update:
  $\mathcal{L}(\theta) = -\sum_{(\mathbf{x}, \tilde{\mathbf{y}}) \in \mathcal{B}_n} \dfrac{\sum_t w_t \log \pi_\theta(\tilde{y}_t|\mathbf{x}, \tilde{y}_{<t})}{\sum_t w_t}, \quad w_t = \sigma\big(\kappa\,[\,\mathrm{sg}[\log \pi_\theta(\tilde{y}_t|\mathbf{x}, \tilde{y}_{<t})] + \gamma\,]\big)$
  Update θ with μ gradient steps on $\mathcal{L}(\theta)$
end for
return trained model π_θ

analyzed prompts. We emphasize that likelihood is used only to rank responses that have already passed the verifier; it is not an additional correctness criterion.

(2) Amplify: success-aware sampling. We place the success $\mathbf{y}^\star$ in context with the prompt, and let the same model generate M new responses:

$$\tilde{\mathbf{y}}_1, \ldots, \tilde{\mathbf{y}}_M \sim \pi_\theta\big(\cdot\,|\mathbf{x}, \mathbf{y}^\star\big). \tag{6}$$

The parameters are unchanged; only the context is different. The verifier checks each new response, and we keep the correct ones. With the demonstration in context, the model succeeds far more often than from the prompt alone (cf. Figure 3). A single rare success therefore turns into several verified responses for the same prompt.

(3) Assimilate: learn from guided successes without the guide. The retained responses were generated using information that will not be available at inference time. SAPL must therefore transfer their useful content into the prompt-only policy. Unweighted imitation treats every target token equally, even when some tokens become much less likely after the demonstration is removed. We use prompt-only token likelihood as a proxy for this mismatch and adopt the surprisal-gated assimilation principle of Pedagogical RL (Chakraborty et al., 2026). For a guided response $\tilde{\mathbf{y}}$, let

$$u_t = \log \pi_\theta(\tilde{y}_t|\mathbf{x}, \tilde{\mathbf{y}}_{<t}), \qquad w_t = \sigma\big(\kappa\,(\mathrm{sg}\big[u_t\big] + \gamma)\big), \tag{7}$$

where σ is the logistic sigmoid and sg stops the gradient. The weight is close to one when the policy already gives the token a reasonable probability, and close to zero when the token is very surprising. The threshold $-\gamma$ decides which log-probabilities count as surprising, and κ decides how sharply the weight switches between the two. The loss is the weighted cross-entropy, normalized within each response:

$$\mathcal{L}_{\mathrm{SAPL}}(\theta) = -\sum_{(\mathbf{x}, \tilde{\mathbf{y}}) \in \mathcal{B}} \frac{\sum_t w_t\, u_t}{\sum_t w_t}, \tag{8}$$

where $\mathcal{B}$ is the set of verified guided responses. With every $w_t = 1$, this is standard cross-entropy. The stop-gradient means the weights only scale the loss; the policy cannot lower the loss by changing them. After assimilation, the updated policy starts the next iteration. SAPL therefore repeatedly converts a discovered on-policy success into a higher-yield guided proposal and then transfers the verifier-accepted responses back into the prompt-only policy. It requires no reference solution, external teacher, or auxiliary context at inference; the only external supervision supplied to the learner is the verifier output. The SAPL approach is summarized in Algorithm 1.

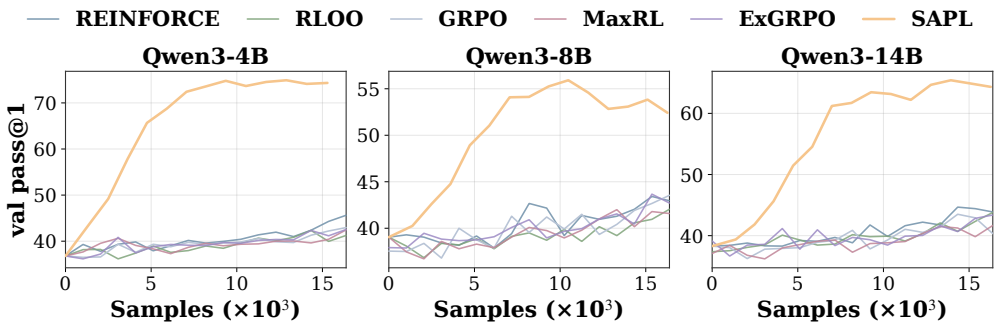

Figure 4: **SAPL versus on-policy RLVR baselines.** Validation pass@1 (accuracy) against the number of sampled responses, over the same budget for every method. All five baselines stay within a few points of where the base model started, at every scale.

4 Experiments

We evaluate SAPL along three axes. First, we test whether it improves learning under a matched response budget relative to verifier-only RLVR methods. Second, we compare it with methods that expose ground-truth answers to the model during training. Third, we analyze how success-guided sampling changes the verified response yield and the resulting learning signal.

4.1 Experimental Setup

Datasets and benchmarks. We use MATH (Hendrycks et al., 2021), DAPO-Math (Yu et al., 2026a) and DeepScaleR (Luo et al., 2025) to source hard prompts for training (see Appendix A.2 for detailed procedure about creating the training and validation splits). To evaluate the performance of the fine-tuned models on hard out-of-distribution math benchmarks, we use MATH-500 (Hendrycks et al., 2021), AIME24, AIME25, AIME26, BeyondAIME, HMMT (Feb 2024, Feb 2025, Nov 2025), CMIMC 2025, OlympiadBench and AMC 2023.

Models and baselines. We fine-tune the instruction-tuned Qwen-3 models at 4B, 8B and 14B (thinking disabled for 8B and 14B models). All methods use the same learning rate, adapter configuration, etc. We compare against five on-policy RLVR methods, which differ only in how the same sparse verifier reward is turned into an advantage: REINFORCE (Xiong et al., 2025) uses the reward directly; GRPO (Shao et al., 2024) normalises it within each group of responses; RLOO (Ahmadian et al., 2024) subtracts a leave-one-out baseline; MaxRL (Tajwar et al., 2026) up-weights successes on prompts that are rarely solved; and ExGRPO (Zhan et al., 2026) replays previously successful responses alongside fresh ones. We further compare against two methods that are given access to external privileged information, i.e., ground-truth answer during training, which SAPL never sees (it only has access to the verifier-interface). Pedagogical RL (Chakraborty et al., 2026)[1] places the ground-truth answer in the model's context, samples solutions from it and trains the model on them using assimilation loss. OPSD (Zhao et al., 2026) keeps the model's ordinary on-policy responses but re-scores them with the answer in context, and pulls the answer-free next-token distribution towards the answer-conditioned one through a token-level divergence. (see Appendix A.3 for hyperparameter values for all algorithms)

4.2 Performance under Verifier-Only Supervision

Figure 4 compares validation performance as a function of the number of generated responses. Across all three model scales, SAPL improves early and continues to make progress, whereas the verifier-only RLVR methods remain substantially closer to their respective base models. Reweighting rare successes, as in MaxRL, or reusing previously observed successes, as in ExGRPO, pro-

[1]We implement a teacher training free version. The original trains the pedagogical teacher with a spike-aware learnability reward before assimilating the samples from trained teacher within the student model; we do not train any separate teacher model first and use the verifier, keeping only the surprisal-gated imitation loss.

Table 1: Test-benchmark accuracy, **pass@8** (%), 32 rollouts per problem, 1024-token completions, RLVR baselines. *Avg.* is the unweighted mean over the eleven benchmarks. Best per column within each model block in **bold**.

	AIME			AMC	Beyond	CMIMC	HMMT			MATH	Olympiad	
Method	'24	'25	'26	'23	AIME	'25	'24	'25	Nov'25	500	Bench	Avg.
Qwen3-4B												
Base	5.8	4.2	3.2	43.8	2.7	1.1	5.3	2.3	4.6	71.2	37.0	16.5
+ REINFORCE	6.1	3.3	4.2	45.8	3.2	2.9	4.2	1.5	4.7	72.2	38.1	16.9
+ GRPO	4.5	3.3	3.2	44.9	2.2	1.5	3.3	3.0	2.6	71.9	37.9	16.2
+ RLOO	7.6	4.2	4.0	43.9	2.6	2.4	4.2	3.0	**5.3**	71.8	37.6	17.0
+ MaxRL	5.9	3.3	3.3	44.2	2.8	1.5	4.8	0.8	4.1	71.5	37.5	16.4
+ ExGRPO	5.9	3.3	3.3	45.5	2.8	1.5	5.7	0.8	4.7	72.0	37.5	16.6
+ Ours	**20.1**	**15.4**	**12.1**	**64.7**	**8.6**	**7.3**	**6.0**	**8.6**	5.0	**82.2**	**52.4**	**25.7**
Qwen3-8B												
Base	14.1	7.5	5.0	51.0	7.1	6.0	4.2	7.1	6.1	81.0	46.9	21.4
+ REINFORCE	18.1	6.7	7.1	56.0	7.2	6.3	5.0	8.3	9.1	81.4	48.4	23.0
+ GRPO	15.1	6.7	6.7	52.9	8.7	5.3	7.6	9.2	7.9	81.3	48.8	22.7
+ RLOO	15.4	7.5	5.6	54.8	7.9	5.9	6.5	7.6	8.9	81.4	48.7	22.7
+ MaxRL	13.9	7.5	7.3	53.2	6.4	4.9	6.8	7.6	6.7	81.8	47.7	22.2
+ ExGRPO	19.5	6.7	8.3	55.3	8.7	7.2	8.0	7.1	7.6	81.7	49.0	23.6
+ Ours	**26.1**	**12.3**	**11.5**	**68.8**	**13.4**	**9.0**	**8.8**	**12.9**	**9.7**	**84.0**	**53.2**	**28.2**
Qwen3-14B												
Base	16.1	7.5	5.7	49.7	4.5	5.8	5.6	4.2	8.6	81.2	47.8	21.5
+ REINFORCE	14.7	7.5	5.2	50.3	5.8	7.2	4.2	5.8	8.8	81.1	48.9	21.8
+ GRPO	17.5	8.1	3.3	49.1	4.6	6.6	4.8	4.1	7.8	81.5	48.4	21.4
+ RLOO	15.7	6.7	5.6	52.0	5.8	6.2	8.1	3.3	9.3	81.5	48.4	22.0
+ MaxRL	16.2	8.3	5.9	52.9	6.0	5.7	5.6	5.0	9.3	81.5	48.5	22.3
+ ExGRPO	16.0	8.6	5.7	52.4	5.3	7.1	4.2	6.7	7.6	81.4	49.0	22.2
+ Ours	**21.1**	**14.0**	**10.9**	**68.5**	**12.7**	**11.4**	**13.6**	**13.5**	**12.1**	**84.4**	**56.4**	**29.0**

vides only limited improvement under this response budget. In contrast, SAPL uses an observed success (see Appendix A.4 for prompt templates) to alter the distribution of subsequent training responses before updating the policy (see Appendix A.6, Figure 9 for training reward and gradient norm curves). The validation improvements transfer to held-out competition benchmarks. As shown in Table 1, SAPL improves average pass@8 over the strongest evaluated verifier-only baseline by 8.7, 4.6, and 6.7 percentage points for Qwen3-4B, 8B, and 14B, respectively. The gains are distributed across benchmarks rather than being driven by a single test set. **Takeaway:** On hard prompts, the bottleneck is the supply of successes, not the estimator applied to them, reweighting or replaying rare successes leaves the baselines near the base model, while using a success to obtain more successes improves both training and out-of-distribution accuracy across scales.

Table 2: Test-benchmark accuracy, **pass@8** (%), 32 rollouts per problem, 1024-token completions, privileged-information baselines. *Avg.* is the unweighted mean over the eleven benchmarks. Best per column within each model block in **bold**.

	AIME			AMC	Beyond	CMIMC	HMMT			MATH	Olympiad	
Method	'24	'25	'26	'23	AIME	'25	'24	'25	Nov'25	500	Bench	Avg.
Qwen3-4B												
Base	5.8	4.2	3.2	43.8	2.7	1.1	5.3	2.3	4.6	71.2	37.0	16.5
+ OPSD	3.8	4.2	4.8	44.7	2.7	1.5	3.3	1.7	2.6	72.1	37.5	16.3
+ Pedagogy	8.1	7.2	3.3	52.3	5.4	7.2	3.3	4.3	**6.5**	77.5	46.0	20.1
+ Ours	**20.1**	**15.4**	**12.1**	**64.7**	**8.6**	**7.3**	**6.0**	**8.6**	5.0	**82.2**	**52.4**	**25.7**
Qwen3-8B												
Base	14.1	7.5	5.0	51.0	7.1	6.0	4.2	7.1	6.1	81.0	46.9	21.4
+ OPSD	16.1	7.5	6.1	52.7	6.8	4.9	6.1	4.1	8.2	80.8	46.7	21.8
+ Pedagogy	22.5	9.6	7.0	63.4	11.2	7.2	6.8	11.9	6.1	83.3	52.4	25.6
+ Ours	**26.1**	**12.3**	**11.5**	**68.8**	**13.4**	**9.0**	**8.8**	**12.9**	**9.7**	**84.0**	**53.2**	**28.2**
Qwen3-14B												
Base	16.1	7.5	5.7	49.7	4.5	5.8	5.6	4.2	8.6	81.2	47.8	21.5
+ OPSD	19.1	8.1	6.3	52.9	5.3	4.5	5.6	7.1	6.2	81.9	49.5	22.4
+ Pedagogy	14.1	11.4	6.1	53.2	9.4	9.9	9.9	7.6	**14.2**	82.2	51.7	24.5
+ Ours	**21.1**	**14.0**	**10.9**	**68.5**	**12.7**	**11.4**	**13.6**	**13.5**	12.1	**84.4**	**56.4**	**29.0**

4.3 Comparison with Answer-Conditioned Training

The previous comparison restricts every method to the verifier-only interface. We next ask how SAPL compares with methods that expose the ground-truth answer to the model during training. This is a stronger information interface: SAPL's generator and training loss receive only the prompt and binary verifier outcomes, whereas OPSD (Zhao et al., 2026) and answer-conditioned assimilation (Chakraborty et al., 2026) may condition directly on the supplied answer.

Table 2 shows that OPSD provides little improvement in average pass@8 over the base models. Generating complete responses with answer-conditioned context is substantially stronger, making answer-conditioned assimilation the more competitive baseline. Nevertheless, SAPL obtains the highest average performance at all three model scales, exceeding this baseline by 5.6, 2.6, and 4.5 percentage points at 4B, 8B, and 14B, respectively (see Appendix A.6 for performance on validation set).

Takeaway: These results show that SAPL can match or exceed the evaluated answer-conditioned alternatives without exposing the answer to the model as generation context or token-level supervision. They should not be interpreted as establishing that a self-generated response is intrinsically more informative than a ground-truth answer, because the methods also differ in how their training responses are generated and used.

4.4 Additional Analysis and Insights: Why does SAPL learn faster?

The preceding results establish that SAPL improves validation and test performance under a limited response budget. We next examine the mechanism in two steps. First, we compare the learning gradients induced by SAPL and GRPO with a high-budget reference gradient. Second, we test whether increasing GRPO's ordinary on-policy sampling budget closes the observed gap.

SAPL induces a gradient better aligned with pass@1 improvement. Let g_{PG} denote the pass@1 gradient in equation (2), estimated using a substantially larger sampling budget than is available during training. For a method-induced training gradient $\widehat{g}$, we measure its directional alignment and its component along the reference gradient:

$$\cos(\widehat{g}, g_{\mathrm{PG}}) = \frac{\langle \widehat{g}, g_{\mathrm{PG}} \rangle}{\|\widehat{g}\| \, \|g_{\mathrm{PG}}\|}, \quad \mathrm{comp}(\widehat{g}, g_{\mathrm{PG}}) = \left\langle \widehat{g}, \frac{g_{\mathrm{PG}}}{\|g_{\mathrm{PG}}\|} \right\rangle. \tag{9}$$

The cosine isolates direction, while the component combines direction and magnitude. Under a first-order approximation and a common step size, a larger positive component predicts greater progress in pass@1. Figure 5 shows that the SAPL gradient is both more closely aligned with g_{PG} and has a larger component along it. GRPO forms its gradient from the few responses found through ordinary on-policy sampling. SAPL instead forms its gradient from a larger verifier-filtered set produced after conditioning on a discovered success. On the analyzed prompts, this difference yields an approximately sixfold increase in the component along the reference direction.

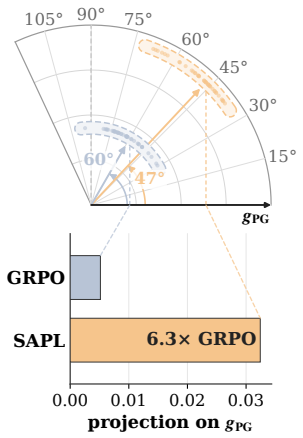

Figure 5: One SAPL update makes more progress on pass@1 than one GRPO: **Top:** angle of update with the pass@1 gradient g_{PG}; dots are individual prompts, arrows their mean. **Bottom:** the projection of each update onto g_{PG} which raises pass@1.

Increasing GRPO's on-policy budget does not close the observed gap. A natural possibility is that GRPO's poorer gradient estimate results only from using too few on-policy responses. Figure 6 tests this explanation over the evaluated budget range. Increasing the number of GRPO responses improves its component along the reference gradient, but does not close the gap with SAPL. This result does not claim that no sufficiently large on-policy budget could match SAPL; rather, it shows that substantially enlarging the group within the tested range does not recover the same learning signal.

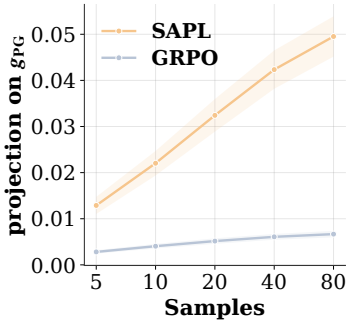

Figure 6: Increasing GRPO's on-policy budget does not close the observed gap. The component of the method-induced gradient along g_{PG} as the number of responses per prompt increases. GRPO improves with budget but remains below SAPL throughout the tested range. Shading denotes bootstrap ± 1 standard error across prompts.

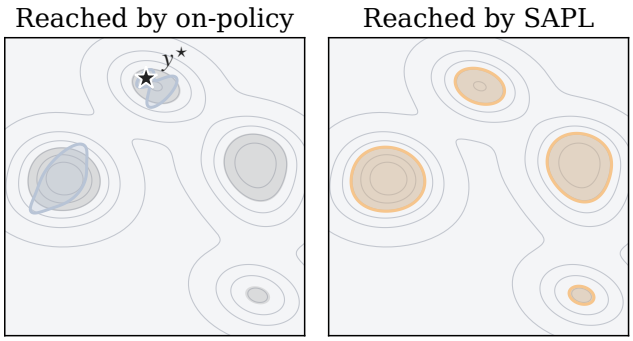

Figure 7: Guided sampling can reach additional correct-response regions. Verifier-accepted responses for one training prompt are embedded in two dimensions. Prompt-only sampling reaches a subset of the observed clusters, whereas SAPL also reaches clusters separated from the selected guide. This qualitative example illustrates a possible behavior of guided sampling; it does not establish broader mode coverage for every prompt.

Success-aware responses are not restatements of the one placed in context. One concern is that guided generation may merely paraphrase the response placed in context. Figure 7 provides a qualitative counterexample. For this prompt, prompt-only correct responses occupy a subset of the observed response clusters, whereas guided sampling also reaches clusters separated from the selected guide. The corresponding textual example in Appendix A.5 shows that the guided response can use a different solution strategy. Because this analysis concerns one problem, we treat it as an illustration of what the mechanism can do, not as evidence that it always increases solution diversity.

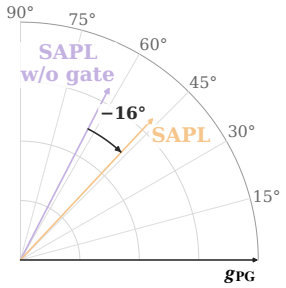

Figure 8: **The gate turns the update closer to the true policy gradient.** Direction of the expected SAPL update with and without the surprisal gate, measured against g_{PG}, a high-sample estimate of the policy gradient at the current parameters.

Assimilation gate steers SAPL gradient direction closer to true policy gradient. Since SAPL trains unguided policy on responses the model generated with access to successful demonstration. Some of their tokens are ones it would almost never write on its own. Therefore, these tokens become highly surprising (see Appendix A.7 for details about surprisal of success-aware samples) for the unguided policy to generate without access to correct response in context. So applying plain cross-entropy gives them full weight to imitate surprising tokens, and they drag the update off course (Figure 8). The gate reduces the weight of such tokens, resulting in a gradient direction that sits 16° closer to g_{PG}, a high-sample estimate of the on-policy gradient.

5 Conclusion

We introduced Success-Aware Policy Learning (SAPL), a discover–amplify–assimilate framework for improving verifier-only RLVR in the low-pass@1 regime. SAPL uses a verified on-policy success as temporary context to acquire additional verifier-accepted responses, and then transfers this experience into the prompt-only policy. Under matched sampling budgets, SAPL achieves a higher average pass@8 than the evaluated RLVR and answer-conditioned baselines across Qwen3-4B, 8B, and 14B, without requiring reference solutions or additional context at inference. These results show that rare successes are valuable not only as targets for policy updates, but also as guidance for acquiring new training experience. A key limitation is that our evaluation focuses on competition mathematics with deterministic binary verifiers, future work should study whether success-aware data acquisition transfers to other verifiable domains, such as code and formal reasoning, and in settings with noisy or partial verification.

AI USE STATEMENT

In this work, we used generative AI tools to assist with implementing research methods. We additionally used generative AI tools to assist with software implementation, drafting and refining scientific figures and diagrams, improving the writing and readability of the paper, and surveying and identifying relevant literature. All AI-assisted code was tested, and figures, literature references, and written content were reviewed and verified by the authors. We take responsibility for the final content of this work, including all text, claims, code, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies methods for improving the sample efficiency of reinforcement learning with verifiable rewards on mathematical reasoning tasks. Our experiments use publicly available mathematics datasets and open-weight language models, and involve no human subjects, personal data, or crowdsourced annotation. SAPL changes how verifier-accepted training data is acquired and assimilated, but it does not introduce new model capabilities beyond those targeted by standard RLVR posttraining. Because SAPL uses only a binary correctness signal, its applicability is limited to domains with reliable automatic verifiers. As with any reward-driven training, applying it with a flawed or exploitable verifier could amplify responses that satisfy the verifier without being genuinely correct, since success-aware sampling multiplies each accepted response. We discuss this limitation, together with the restriction of our evaluation to competition mathematics with deterministic verifiers, in Section 5.

REPRODUCIBILITY STATEMENT

We provide the information needed to reproduce our methods and experiments throughout the paper and appendix. The problem setting and the connection between the baseline-free policy gradient and learning from verified successes are formalized in Section 2. SAPL is specified mathematically in Section 3 (Equations 5- 8) and algorithmically in Algorithm 1. The experimental setup, baselines, and evaluation benchmarks are described in Section 4.1. Appendix A.2 details the construction of the hard training and validation splits, Appendix A.3 reports all training configurations and hyperparameters for SAPL and every baseline, and Appendix A.4 gives the exact system and success-aware prompt templates. All experiments use publicly available datasets (MATH, DAPO-Math-17k, and DeepScaleR) and publicly available models (Qwen3-4B, 8B, and 14B). We intend to release our implementation, data splits, and evaluation scripts upon publication.

REFERENCES

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting reinforce-style optimization for learning from human feedback in llms. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12248–12267, 2024. 6

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020. 2

Souradip Chakraborty, Noah Ziems, Furong Huang, Meng Jiang, Amrit Singh Bedi, and Omar Khattab. Pedagogical rl: Teaching models to teach themselves from privileged information. 2026. https://noahziems.com/pedagogical-rl. 2, 5, 6, 8, 13, 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. 2, 13

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021. 6, 13

Jonas Hübotter, Frederike Lübeck, Lejs Behric, Anton Baumann, Marco Bagatella, Daniel Marta, Ido Hakimi, Idan Shenfeld, Thomas Kleine Buening, Carlos Guestrin, et al. Reinforcement learning via self-distillation. *arXiv preprint arXiv:2601.20802*, 2026. 2, 13

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Li Erran Li, Raluca Ada Popa, et al. Deepscaler: Surpassing o1-preview with a 1.5 b model by scaling rl. *Notion Blog*, 3(5), 2025. 6, 13

Deng Qiyuan, Kehai Chen, Min Zhang, and Zhongwen Xu. Hipo: Self-hint policy optimization for rlvr. In *The Fourteenth International Conference on Learning Representations*, 2026. 2, 13

Daniel Russo. Success conditioning as policy improvement: The optimization problem solved by imitating success. *arXiv preprint arXiv:2601.18175*, 2026. 2, 4, 13

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. 2, 6, 13

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. *arXiv preprint arXiv:2312.06585*, 2023. 13

Fahim Tajwar, Guanning Zeng, Yueer Zhou, Yuda Song, Daman Arora, Yiding Jiang, Jeff Schneider, Ruslan Salakhutdinov, Haiwen Feng, and Andrea Zanette. Maximum likelihood reinforcement learning. *arXiv preprint arXiv:2602.02710*, 2026. 2, 6, 13

Vladimir Vapnik and Akshay Vashist. A new learning paradigm: Learning using privileged information. *Neural networks*, 22(5-6):544–557, 2009. 13

Wei Xiong, Jiarui Yao, Yuhui Xu, Bo Pang, Lei Wang, Doyen Sahoo, Junnan Li, Nan Jiang, Tong Zhang, Caiming Xiong, et al. A minimalist approach to llm reasoning: from rejection sampling to reinforce. *arXiv preprint arXiv:2504.11343*, 2025. 6

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *Advances in Neural Information Processing Systems*, 38:113222–113244, 2026a. 6, 13

Weichen Yu, Xiaomin Li, Yizhou Zhao, Xiaoze Liu, Ruowang Zhang, Haixin Wang, Yinyi Luo, Chen Henry Wu, Gaurav Mittal, Matt Fredrikson, et al. Multi-rollout on-policy distillation via peer successes and failures. *arXiv preprint arXiv:2605.12652*, 2026b. 2, 13

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488, 2022. 13

Runzhe Zhan, Yafu Li, Zhi Wang, Xiaoye Qu, Dongrui Liu, Jing Shao, Derek Wong, and Yu Cheng. Exgrpo: Learning to reason from experience. In *International Conference on Learning Representations*, volume 2026, pp. 12810–12846, 2026. 2, 6

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. 6, 8, 13

A Appendix

A.1 Detailed Context of Related Work

Reinforcement learning from verifier rewards. Reinforcement learning with verifiable rewards (RLVR) improves reasoning models using automatically checked outcomes, with GRPO enabling critic-free, group-relative policy updates (Shao et al., 2024; Guo et al., 2025). MaxRL revisits the optimization objective through sampling-based approximations to maximum likelihood (Tajwar et al., 2026), while self-training methods such as STaR and ReST^{EM} learn from correctness-filtered, model-generated solutions (Zelikman et al., 2022; Singh et al., 2023). Complementing these algorithmic approaches, Russo (2026) characterizes exact success conditioning as a trust-region policy-improvement operator. These connections motivate learning from verified responses, but leave open how best to acquire useful successful experience under a limited sampling budget. Our work focuses on the opportunity after a success is discovered: using its content to guide additional data acquisition before updating the policy.

Success-guided exploration and self-distillation. Several methods already exploit successful responses beyond direct reinforcement. HiPO reuses prefixes of successful policy rollouts as hints for further exploration and group-relative learning (Qiyuan et al., 2026). SDPO uses successful peer rollouts as feedback for dense supervision on student-generated attempts (Hübotter et al., 2026), while MOPD extends this conditioning to multiple successful and failed trajectories (Yu et al., 2026b). SDPO also evaluates supervised fine-tuning on successful generations from its feedback-conditioned self-teacher. Success-Aware Policy Learning (SAPL) studies a specific combination of these sampling and learning perspectives: a complete verified response guides fresh full-response generation, the resulting candidates are independently verified, and unguided policy-compatible responses are selected for learning without the guidance context. Our emphasis is on coupling this post-success expansion with trajectory- and token-level compatibility.

Privileged and pedagogical learning. Using auxiliary context during training but not inference connects our approach to learning using privileged information (Vapnik & Vashist, 2009). OPSD uses a solution-conditioned self-teacher to provide token-level supervision on student rollouts (Zhao et al., 2026). Pedagogical RL additionally optimizes a privileged self-teacher to generate correct, student-learnable trajectories, which the student absorbs through surprisal-gated assimilation (Chakraborty et al., 2026). We adopt its assimilation principle, but construct the guidance from the policy's own online verified successes rather than supplied reference answers or solutions. Instead of separately optimizing a pedagogical teacher, SAPL selects compatible verified responses and weights their tokens under the prompt-only policy. The guidance response is absent from both the student's training input and its inference context.

A.2 Hard task creation

We start from three public mathematics corpora: MATH (Hendrycks et al., 2021), DAPO-Math-17k (Yu et al., 2026a) and DeepScaleR (Luo et al., 2025) to source prompts for training and validation monitoring. Before finding the set of hard prompts well-tailored for each model separately, we clean the datasets to make sure training and evaluation results are fair. For example, problems that appear in any of our evaluation benchmarks and duplicates are removed. Then for every model separately, we draw $G = 16$ responses for every problem and group the problems by their success rate (or pass@1). This gives a subset of prompts for training with an average pass@1 of 3%, and a validation set of moderate difficulty with an average pass@1 of 37%.

A.3 Training configurations

All runs share the same sampling geometry: $B = 16$ prompts per step and $K = 16$ on-policy rollouts per prompt, i.e. 256 on-policy rollouts per generation, at `max_tokens` $= 1024$, temperature $= 0.8$, top-$p = 0.95$. All arms train LoRA adapters ($r = 64$, $\alpha = 128$, dropout 0.05) with `adamw_8bit`, weight decay 0.01, a constant learning rate, gradient clipping at 0.3 and no warmup. The learning rate is $1\mathrm{e}{-6}$ for Qwen3-4B and $2\mathrm{e}{-6}$ for Qwen3-8B and Qwen3-14B. μ denotes the number of optimizer steps taken on one generation, so the number of generations is steps$/\mu$. Methods that also draw success-aware responses use a second group size M.

A.3.1 SAPL

Model	Steps	μ	Generations	K	M	Per-device batch	Eval every
Qwen3-4B	384	6	64	16	4	32	24
Qwen3-8B	256	4	64	16	4	32	16
Qwen3-14B	256	4	64	16	4	32	16

Surprisal-gated loss with $\kappa = 1.0$ and $\gamma = -4$.

A.3.2 RLVR BASELINES (GRPO, REINFORCE, RLOO, MAXRL, EXGRPO)

Model	Steps	μ	Generations	K	Per-device batch	Eval every
Qwen3-4B	64	1	64	16	32	4
Qwen3-8B	64	1	64	16	32	4
Qwen3-14B	64	1	64	16	16	4

All five arms share this geometry. GRPO uses $\epsilon = 0.2$, $\beta = 0$, group-scaled rewards and token-level importance sampling; REINFORCE, RLOO, MaxRL and GRPO differ only in their advantage estimator. ExGRPO adds a replay buffer with $\rho = 0.5$, $\beta = 0.1$ and a pool of 8.

A.3.3 OPSD

Model	Steps	μ	Generations	G_o	Per-device batch	Eval every
Qwen3-4B	64	1	64	16	8	4
Qwen3-8B	64	1	64	16	8	4
Qwen3-14B	64	1	64	16	8	4

On-policy self-distillation against an EMA copy of the policy (decay 0.95), JSD loss at distillation temperature 1.0, $\beta = 0$, token clipping at 0.05. The smaller per-device batch reflects holding two models in memory.

A.3.4 PEDAGOGICAL RL

Model	Steps	μ	Generations	K	M	Per-device batch	Eval every
Qwen3-4B	384	6	64	16	16	32	24
Qwen3-8B	256	4	64	16	16	32	16
Qwen3-14B	256	4	64	16	16	32	16

A.4 PROMPT TEMPLATES USED FOR RLVR BASELINES AND SAPL

This section provides the prompt templates used during on-policy sampling and success-aware sampling.

A.4.1 SYSTEM PROMPT

The following prompt is used as system prompt for all math experiments

```
Please reason step by step, and put your final answer
within \boxed{}.
```

A.4.2 SAPL PROMPT, QWEN3 MODELS

The following prompt is carefully engineered for Qwen3 models during success-aware sampling in SAPL.

```
You are given a problem and a hint.

Your task is to carefully study the hint, use it to guide
your reasoning, and solve the problem completely. Ensure
your solution is logically consistent and leads to a
complete and correct final answer.

**Important**: Show your reasoning step-by-step, and
clearly present the final answer using LaTeX-style
`\boxed{}` notation.

**Do not mention the hint, reference it, or acknowledge
that you were given one. Your response should read as if
you solved the problem entirely on your own.**

Problem:
{problem}

Hint:
{hint}

Solve the problem. Make sure your final answer is written
as: \boxed{your_answer_here}
```

A.5 Qualitative example of on-policy success and success-aware sample

The following example shows success-aware sampling during amplification stage does not merely copy or paraphrase the maximum likelihood success but produces a different method to solve the same problem.

Example

Prompt
Let a sequence $b_1, b_2, \ldots$ be defined as $b_1 = 1$, $b_2 = 2$, and $b_{n+1} = b_n b_{n-1}$. Calculate b_{20}.

Ground truth: 2^{4181}

On-policy response:

We are given a sequence defined by $b_1 = 1$, $b_2 = 2$, $b_{n+1} = b_n \cdot b_{n-1}$ for $n \geq 2$. We are to compute b_{20}.

Step 1: Understand the recurrence. This is a non-linear recurrence (multiplicative), where each term is the product of the two preceding terms. Let's compute the first few terms to see if we can find a pattern.

Step 2: Compute terms step by step.

- $b_1 = 1, \quad b_2 = 2$
- $b_3 = b_2 \cdot b_1 = 2 \cdot 1 = 2$
- $b_4 = b_3 \cdot b_2 = 2 \cdot 2 = 4$
- $b_5 = b_4 \cdot b_3 = 4 \cdot 2 = 8$
- $b_6 = b_5 \cdot b_4 = 8 \cdot 4 = 32$
- $b_7 = b_6 \cdot b_5 = 32 \cdot 8 = 256$
- $b_8 = b_7 \cdot b_6 = 256 \cdot 32 = 8192$
- $b_9 = b_8 \cdot b_7 = 8192 \cdot 256$

Let's compute that: $8192 \cdot 256$. Note $8192 = 2^{13}$ and $256 = 2^8$, so $8192 \cdot 256 = 2^{13} \cdot 2^8 = 2^{21} = 2097152$. So $b_9 = 2^{21}$. Let's continue:

- $b_{10} = b_9 \cdot b_8 = 2^{21} \cdot 2^{13} = 2^{34}$
- $b_{11} = b_{10} \cdot b_9 = 2^{34} \cdot 2^{21} = 2^{55}$

- $b_{12} = b_{11} \cdot b_{10} = 2^{55} \cdot 2^{34} = 2^{89}$
- $b_{13} = b_{12} \cdot b_{11} = 2^{89} \cdot 2^{55} = 2^{144}$
- $b_{14} = b_{13} \cdot b_{12} = 2^{144} \cdot 2^{89} = 2^{233}$
- $b_{15} = b_{14} \cdot b_{13} = 2^{233} \cdot 2^{144} = 2^{377}$
- $b_{16} = b_{15} \cdot b_{14} = 2^{377} \cdot 2^{233} = 2^{610}$
- $b_{17} = b_{16} \cdot b_{15} = 2^{610} \cdot 2^{377} = 2^{987}$
- $b_{18} = b_{17} \cdot b_{16} = 2^{987} \cdot 2^{610} = 2^{1597}$
- $b_{19} = b_{18} \cdot b_{17} = 2^{1597} \cdot 2^{987} = 2^{2584}$
- $b_{20} = b_{19} \cdot b_{18} = 2^{2584} \cdot 2^{1597} = 2^{4181}$

Final Answer: $\boxed{2^{4181}}$. This is the value of b_{20}.

SAPL response:

We are given a sequence defined by $b_1 = 1$, $b_2 = 2$, $b_{n+1} = b_n \cdot b_{n-1}$ for $n \geq 2$. We are to compute b_{20}.

Step 1: Analyze the recurrence. The recurrence is multiplicative: each term is the product of the two preceding terms. This is a non-linear recurrence, and direct computation of large terms would be cumbersome without recognizing a pattern. We proceed by computing the first few terms and look for a pattern in their exponents when expressed as powers of 2. Note that all terms after b_1 and b_2 grow very quickly, and in fact, from the initial values $b_1 = 1 = 2^0$ and $b_2 = 2 = 2^1$. Let us compute the values step by step and express each as a power of 2.

Step 2: Compute terms and track exponents. Let $b_n = 2^{e_n}$, so we define e_n such that $b_n = 2^{e_n}$. We'll find a recurrence for e_n. We start: $b_1 = 1 = 2^0 \to e_1 = 0$ and $b_2 = 2 = 2^1 \to e_2 = 1$. Now, for $n \geq 2$, $b_{n+1} = b_n \cdot b_{n-1}$, so in exponents

$$e_{n+1} = e_n + e_{n-1}\,.$$

This is the Fibonacci recurrence! Thus the exponents e_n satisfy $e_1 = 0$, $e_2 = 1$, $e_{n+1} = e_n + e_{n-1}$ for $n \geq 2$. This is exactly the Fibonacci sequence, with initial values $0, 1, 1, 2, 3, 5, \ldots$ Let's list the first few e_n:

- $e_1 = 0, \quad e_2 = 1$
- $e_3 = e_2 + e_1 = 1 + 0 = 1$
- $e_4 = e_3 + e_2 = 1 + 1 = 2$
- $e_5 = e_4 + e_3 = 2 + 1 = 3$
- $e_6 = 3 + 2 = 5, \quad e_7 = 5 + 3 = 8, \quad e_8 = 8 + 5 = 13$
- $e_9 = 13 + 8 = 21, \quad e_{10} = 21 + 13 = 34$
- $e_{11} = 34 + 21 = 55, \quad e_{12} = 55 + 34 = 89$
- $e_{13} = 89 + 55 = 144, \quad e_{14} = 144 + 89 = 233$
- $e_{15} = 233 + 144 = 377, \quad e_{16} = 377 + 233 = 610$
- $e_{17} = 610 + 377 = 987, \quad e_{18} = 987 + 610 = 1597$
- $e_{19} = 1597 + 987 = 2584, \quad e_{20} = 2584 + 1597 = 4181$

Therefore $b_{20} = 2^{e_{20}} = 2^{4181}$.

Final Answer: $\boxed{2^{4181}}$

Both correct. On-policy response multiplies term by term and converts to powers of 2 only when forced to evaluate $8192 \cdot 256$. SAPL response substitutes $b_n = 2^{e_n}$ first, which turns the recurrence into Fibonacci.

A.6 TRAINING CURVES

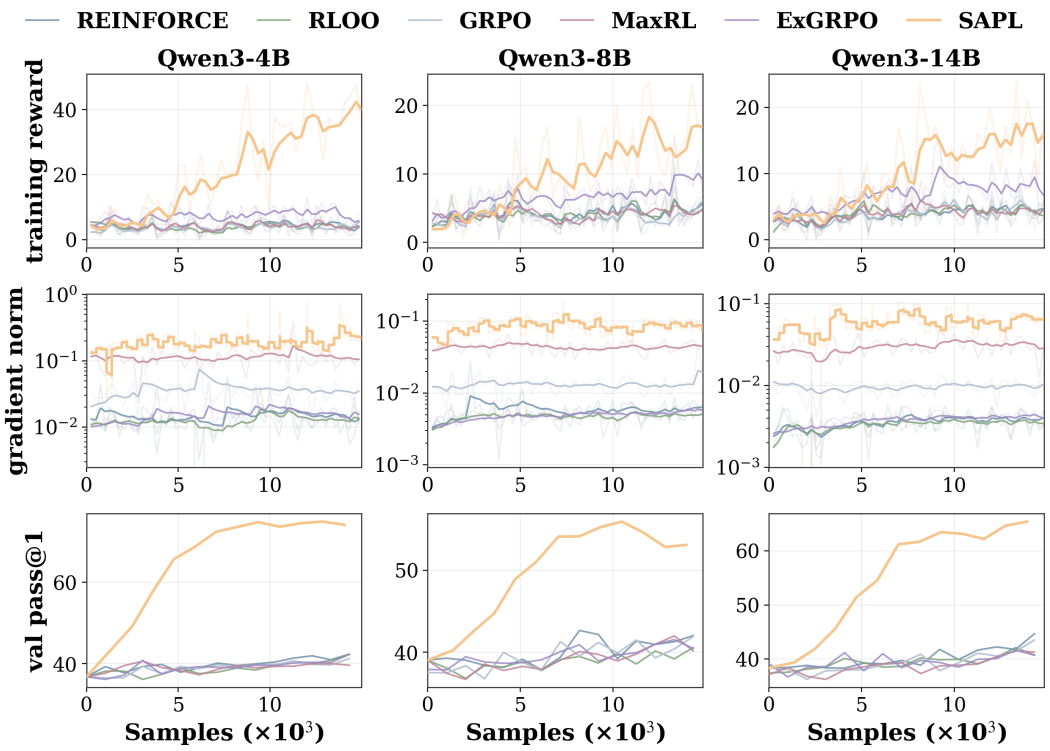

Figure 9: **SAPL turns the sampling budget into learning; the RLVR baselines do not.** Rows are training reward, gradient norm (log scale) and validation pass@1; columns are the three model sizes. The x axis is the cumulative number of rollouts drawn, so the arms are compared at equal sampling cost. At every size SAPL's training reward and validation pass@1 climb steadily, while the baselines stay close to where the base model started and their gradient norms stay smaller than SAPL: on data this hard, most prompts return no usable reward signal.

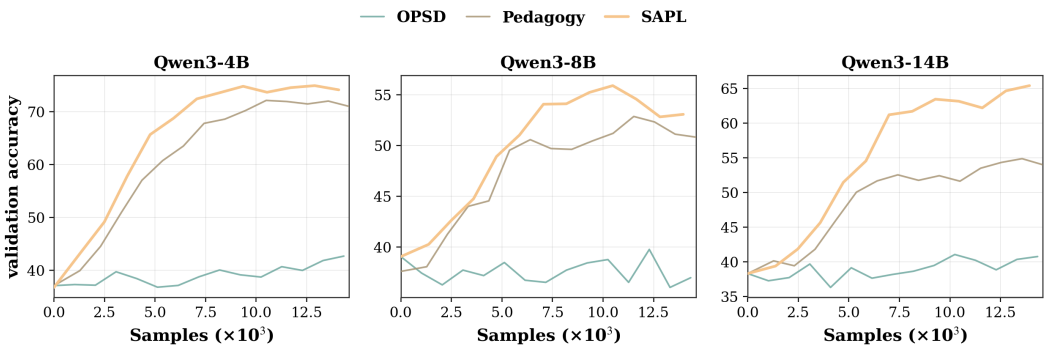

Figure 10: **What matters is conditioning on a success, not distilling the model into itself.** Validation pass@1 for the three model sizes. The x axis is the cumulative number of rollouts drawn, so the arms are compared at equal sampling cost. OPSD generates samples from unguided policy model and only re-scores them against a privileged policy that has been shown the reference solution. OPSD still generates majority of incorrect responses for hard tasks despite getting token-level supervision from privileged policy, slowing down the learning speed. Pedagogical RL uses privileged policy to generate better samples but still requires access to external ground truth which is not always available.

A.7 Additional Insights: Surprisal warrants assimilation gate

The gate determines how much weight each token of a success-aware response receives, based on how likely the policy is to produce it without guidance. To see what it is deciding about, we use the *surprise gap* of Chakraborty et al. (2026) which checks, at each position, how far the response strayed from what the policy itself would have written.

$$d_t \;=\; \log \pi_\theta(\hat{y}_t \mid \mathbf{x}, \tilde{y}_{<t}) \;-\; \log \pi_\theta(\tilde{y}_t \mid \mathbf{x}, \tilde{y}_{<t}), \qquad \hat{y}_t \;=\; \arg\max_y \pi_\theta(y \mid \mathbf{x}, \tilde{y}_{<t}). \tag{10}$$

It is zero wherever the response agrees with the policy's own choice, and grows with how strongly the policy preferred something else.

The gate reads a token's own log-probability u_t, not the gap. The two are linked by

$$u_t = m_t - d_t, \qquad m_t = \max_y \log \pi_\theta(y \mid \mathbf{x}, \tilde{y}_{<t}) \le 0, \tag{11}$$

so $u_t \le -d_t$: a large gap forces a small probability, and therefore a small weight. Every token in the right tail of Figure 11 is one the gate suppresses. We report the gap rather than u_t because u_t alone cannot separate the two reasons a token can be unlikely. Where the policy is simply uncertain, every continuation has a low u_t, including the one the policy would have sampled itself. The gap is zero exactly when the token is the policy's own choice, so it counts only the tokens that were genuinely imported.

Figure 11 shows two very different pictures. An on-policy response is almost entirely what the policy would have produced anyway; its gaps pile up at zero. A success-aware response is not. It carries a steady population of tokens the policy would not have reached, and the largest gaps have no on-policy counterpart at all. Those tokens are the new part of the response, and also the part the policy has practically no support for. Down-weighting them is what keeps the update from being dominated by tokens the policy has no support for, which is why the gate brings the gradient direction induced by the success-aware samples closer to the on-policy gradient (Figure 8).

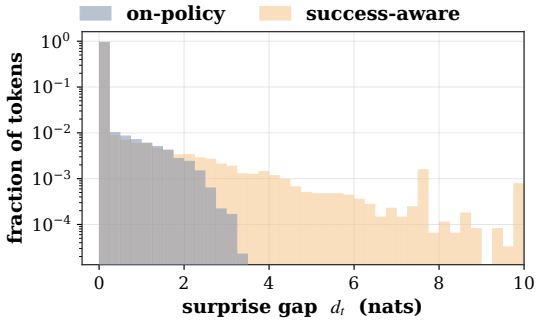

Figure 11: **Success-aware responses carry tokens the policy would not have produced on its own.** Distribution of the per-token surprise gap for a math problem, as a fraction of each population's tokens, on a log scale.