SELF-DISTILLED REASONER: ON-POLICY SELF-DISTILLATION FOR LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Knowledge distillation improves large language model (LLM) reasoning by compressing the knowledge of a teacher LLM to train smaller LLMs. On-policy distillation advances this approach by having the student sample its own trajectories while a teacher LLM provides dense token-level supervision, addressing the distribution mismatch between training and inference in off-policy distillation methods. However, on-policy distillation typically requires a separate, often larger, teacher LLM and does not explicitly leverage ground-truth solutions available in reasoning datasets. Inspired by the intuition that a sufficiently capable LLM can rationalize external privileged reasoning traces and teach its weaker self (i.e., the version without access to privileged information), we introduce *On-Policy Self-Distillation* (OPSD), a framework where a single model acts as both teacher and student by conditioning on different contexts. The teacher policy conditions on privileged information (e.g., verified reasoning traces) while the student policy sees only the question; training minimizes the per-token divergence between these distributions over the student's own rollouts. We demonstrate the efficacy of our method on multiple mathematical reasoning benchmarks, achieving 4-8× token efficiency compared to reinforcement learning methods such as GRPO and superior performance over off-policy distillation methods.

1 INTRODUCTION

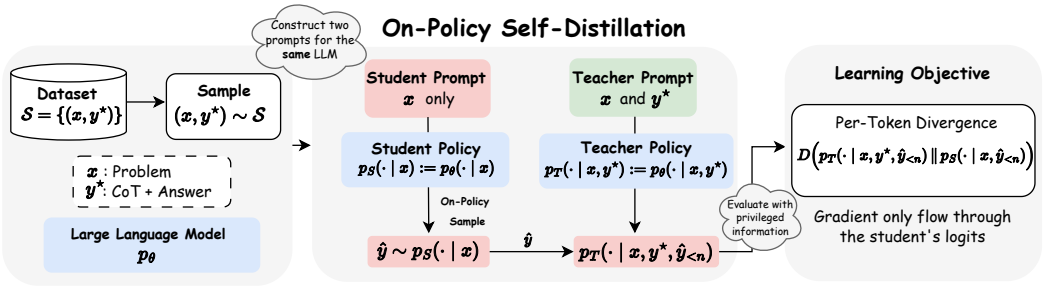

Figure 1: **Overview of On-Policy Self-Distillation (OPSD):** Given a reasoning dataset $\mathcal{S} = \{(x_i, y_i^\star)\}_{i=1}^N$, we instantiate two policies from the same LLM: a *student policy* $p_S(\cdot \mid x)$ and a *teacher policy* $p_T(\cdot \mid x, y^\star)$. The student generates an on-policy response $\hat{y} \sim p_S(\cdot \mid x)$. Both policies then evaluate this trajectory to produce next-token distributions $p_S(\cdot \mid x, \hat{y}_{<n})$ and $p_T(\cdot \mid x, y^\star, \hat{y}_{<n})$ at each step n. The learning objective minimizes the per-token divergence $D(p_T \| p_S)$ along the student's rollout. Crucially, gradients backpropagate only through the student's logits, allowing the model to self-distil.

Recent advances in large language models (LLMs) have demonstrated impressive capabilities in reasoning and instruction following. Achieving these capabilities during post-training typically relies on reinforcement learning methods such as Reinforcement Learning with Verifiable Rewards (RLVR) (e.g., GRPO (Shao et al., 2024; Guo et al., 2025; Team et al., 2025; Rastogi et al., 2025; Yu et al., 2025)), supervised fine-tuning (SFT) on high-quality reasoning datasets (Guha et al., 2025; Team et al., 2025; Xiaomi, 2026), or knowledge distillation, where recent work has shown that

distillation from advanced teacher models can outperform RL in both performance and training efficiency (Yang et al., 2025; Xiaomi, 2026; Lu & Lab, 2025).

Despite their respective successes, each approach has inherent limitations. RLVR suffers from inefficiencies including: (1) sampling a group of responses per prompt is computationally expensive and can introduce high variance in estimating the true value function; moreover, when all samples are either correct or incorrect, the gradient signal vanishes (Yu et al., 2025; Zhao et al., 2025); and (2) the reward signal is sparse and uniformly applied across all tokens in the generated output, neglecting fine-grained token-level feedback. Supervised fine-tuning suffers from exposure bias and weaker generalization (Agarwal et al., 2024; Chu et al., 2025). Traditional knowledge distillation provides dense token-level supervision from a teacher model but relies on off-policy data (Hinton et al., 2015). Recent advances in on-policy distillation—where a student model samples its own trajectories while a teacher policy provides dense token-level supervision—have demonstrated superior sample efficiency by combining the distributional realism of on-policy training with dense feedback (Agarwal et al., 2024; Lu & Lab, 2025).

While on-policy distillation has shown strong performance, it relies on a distinct teacher model to supervise the student. Given that modern LLMs already exhibit strong reasoning capabilities, we ask this research question: *can a model effectively serve as its own teacher through self-distillation?* Our approach is inspired by human learning: after solving a problem incorrectly, a student can examine the correct solution, rationalize its steps, and identify where their reasoning failed. Prior work has shown that for LLMs, evaluation is often easier than generation (Sun et al., 2024; Naor, 1996). We hypothesize that *rationalization*—explaining a given correct answer—is similarly easier than generation. Motivated by this, we instantiate both the teacher and student policies from a single LLM. The teacher policy is provided with privileged information $y^\star$, such as the ground-truth answer or a reference chain-of-thought, while the student policy conditions only on the problem x. Concretely, the teacher policy $p_T(\cdot \mid x, y^\star)$ conditions on both the problem and the privileged answer, whereas the student policy $p_S(\cdot \mid x)$ observes only the problem. We preserve the on-policy training paradigm by sampling trajectories $\hat{y}$ exclusively from the student policy, which then receives dense, token-level supervision from the privileged teacher policy.

We therefore propose **On-Policy Self-Distillation (OPSD)**, a framework in which a single model plays both teacher and student roles. The student samples its own trajectories $\hat{y} \sim p_S(\cdot \mid x)$; we then compute the per-token divergence between the student and teacher distributions and minimize it over the student's own rollouts. This formulation (i) uses on-policy supervision (the student's own trajectories), (ii) provides dense per-token feedback, (iii) exploits ground-truth solutions $y^\star$, and (iv) requires no separate teacher model. The learning process is captured by the loss

$$\mathcal{L}_{\mathrm{OPSD}}(\theta) = \mathbb{E}_{(x,y^\star)\sim\mathcal{S}}\, \mathbb{E}_{\hat{y}\sim p_S(\cdot|x)} \sum_{n=1}^{|\hat{y}|} D\Big(p_T(\cdot \mid x, y^\star, \hat{y}_{<n}) \;\Big\|\; p_S(\cdot \mid x, \hat{y}_{<n})\Big). \tag{1}$$

In summary, our contributions are as follows:

- We introduce On-Policy Self-Distillation, a novel framework that enables a single model to act as both teacher and student, leveraging ground-truth answers to provide dense token-level supervision on student rollouts.
- We evaluate OPSD on four competition-level mathematical reasoning tasks, demonstrating that it outperforms both RLVR (e.g., GRPO) and supervised fine-tuning baselines.
- We show that OPSD achieves better performance with up to $8\times$ improved token efficiency than GRPO.
- We analyze the impact of model scale, finding that moderate model capacity is necessary for successful self-distillation. We further compare different divergence objectives and analyze the effect of student generation length.

2 Background

2.1 Knowledge Distillation for Autoregressive Large Language Models

Knowledge distillation transfers knowledge from a larger teacher model to a smaller student model by training the student to mimic the teacher's behavior (Hinton et al., 2015; Kim & Rush, 2016; Sanh

	SFT/Off-Policy Distillation	GRPO	On-Policy Distillation	On-Policy Self-Distillation (Ours)
On-Policy Data	✗	✓	✓	✓
Dense Learning Signal	✓	✗	✓	✓
Low Sampling Cost	✓	✗	✓	✓
No External Teacher	✓	✓	✗	✓

Table 1: Comparison of training methods for reasoning tasks. On-Policy Self-Distillation (OPSD) combines the advantages of on-policy training with dense feedback without requiring an external teacher model.

et al., 2019). The core insight is that the teacher's soft probability distribution over classes contains richer information than hard labels alone, as it reveals the teacher's learned similarities between classes. For auto-regressive language models, given a dataset $\mathcal{S} = \{(x, y^\star)\}$ where x denotes an input and $y^\star$ is the corresponding reference output, both teacher p_T and student p_S define token-level distributions over vocabulary $\mathcal{V}$. Traditional supervised distillation minimizes a divergence D between teacher and student distributions averaged over a fixed dataset:

$$\mathcal{L}_{\text{Supervised Distillation}}(\theta) = \mathbb{E}_{(x,y)\sim\mathcal{S}}[D(p_T \| p_S)(y|x)], \tag{2}$$

where $D(p_T \| p_S)(y|x) = \frac{1}{|y|}\sum_{n=1}^{|y|} D(p_T(\cdot|y_{<n}, x) \| p_S(\cdot|y_{<n}, x))$ measures per-token discrepancy. However, this off-policy approach suffers from distribution mismatch: the student encounters different partial sequences $y_{<n}$ during auto-regressive generation at inference than those seen during training on the fixed dataset, leading to compounding errors. On-policy distillation (Agarwal et al., 2024; Lu & Lab, 2025; Xu et al., 2024a) addresses this by training the student on its own generated sequences $\hat{y} \sim p_S(\cdot|x)$, obtaining dense token-level feedback from the teacher on these on-policy samples:

$$\mathcal{L}_{\text{On-Policy Distillation}}(\theta) = \mathbb{E}_{x\sim\mathcal{S}}[\mathbb{E}_{\hat{y}\sim p_S(\cdot|x)}[D(p_T \| p_S)(\hat{y}|x)]]. \tag{3}$$

This approach connects distillation to imitation learning (Ross et al., 2011), where the student iteratively improves by learning from the teacher's guidance on its own outputs, combining the on-policy relevance of reinforcement learning with the dense reward signal of supervised learning, thereby mitigating exposure bias while maintaining computational efficiency.

2.2 Reinforcement Learning with Verifiable Rewards

Reinforcement learning with verifiable rewards (RLVR) has emerged as a popular approach for post-training large language models, particularly on tasks with easily verifiable outcomes such as mathematics and coding, using algorithms like Proximal Policy Optimization (PPO) (Schulman et al., 2017) and Group Relative Policy Optimization (GRPO) (Shao et al., 2024).

GRPO trains by sampling a group of G responses $\{o_1, o_2, \ldots, o_G\}$ from the current policy π_θ for each problem x. Each response o_i receives a binary reward $r_i \in \{0, 1\}$ indicating correctness. The method then assigns advantages to all tokens $k = 1, \ldots, |o_i|$ within response o_i using a group-normalized reward: $A_i = \frac{r_i - \text{mean}(\{r_j\}_{j=1}^G)}{\text{std}(\{r_j\}_{j=1}^G)}$. This formulation can be understood through the value function lens: $\text{mean}(\{r_j\}_{j=1}^G)$ serves as a G-sample Monte Carlo estimate of the value function $V(x)$, while the sparse binary reward r_i represents the (undiscounted) state-action value $Q(x, o_i)$. Critically, all tokens within a response share the same advantage, as the reward signal is provided only at the sequence level. The GRPO objective incorporates a clipped surrogate loss to moderate policy updates, along with a reverse KL penalty to prevent excessive deviation from a reference policy:

$$\begin{aligned}\mathcal{L}_{\text{GRPO}}(\theta) = \mathbb{E}_{\substack{x\sim\mathcal{S} \\ o_1,\ldots,o_G\sim\pi_\theta(\cdot|x)}} \Bigg[\frac{1}{G}\sum_{i=1}^{G} \frac{1}{|o_i|}\sum_{n=1}^{|o_i|} \min\left(\rho_i^n A_i, \text{clip}\left(\rho_i^n, 1-\varepsilon, 1+\varepsilon\right) A_i\right) \\ - \beta D_{\text{KL}}[\pi_\theta(\cdot|x) \| \pi_{\text{ref}}(\cdot|x)] \Bigg]\end{aligned} \tag{4}$$

Student Prompt

Problem: Find the derivative of $f(x) = 3x^2 + 2x - 5$ at $x = 2$

Answer:

Teacher Prompt

Problem: Find the derivative of $f(x) = 3x^2 + 2x - 5$ at $x = 2$

Here is a reference solution:
First find $f'(x) = 6x + 2$, then evaluate at $x = 2$: $f'(2) = 6(2) + 2 = 14$

After understanding the reference solution, please try to solve this problem using your own approach below:

Answer:

Figure 2: **Prompt example for student and teacher policies.** Both policies share the same parameters θ but differ in conditioning context. The teacher receives the ground-truth solution $y^\star$ as privileged information before generation. To ensure a natural transition before evaluating the student's rollout, the teacher is prompted to rationalize and generate its own solution. Note that the teacher won't be generating tokens—rationalization is done implictly through one forward pass.

where $\rho_i^n = \frac{\pi_\theta(o_i^n|x,o_i^{<n})}{\pi_{\theta_{\text{old}}}(o_i^n|x,o_i^{<n})}$ is the importance ratio, $\pi_{\theta_{\text{old}}}$ is the policy before the update, and ε controls the clipping range.

While RLVR methods have demonstrated strong empirical performance, they face two key limitations: (1) the reward signal is sparse, providing only sequence-level feedback rather than token-level guidance on where errors occur, and (2) when all sampled responses receive identical rewards (all correct or all incorrect), the advantages become zero, preventing any policy update despite the computational cost of sampling.

3 METHODS

3.1 LEARNING FROM VERIFIABLE REASONING DATASET

We consider a dataset of problem-solution pairs $\mathcal{S} = \{(x_i, y_i^\star)\}_{i=1}^N$, where each x_i denotes a problem and $y_i^\star$ is the corresponding reference solution, which may include chain-of-thought reasoning. For brevity, we omit the sample index i and use $(x, y^\star)$ to denote a generic sample from the dataset. We can exploit learning signals from this dataset from different ways: Standard supervised fine-tuning (SFT) on $\mathcal{S}$ can be viewed as off-policy distillation/imitation learning using expert trajectories, but it suffers from distribution mismatch between training and inference. Reinforcement learning from verifiable rewards (RLVR), such as GRPO, addresses this by optimizing on-policy samples and assigning binary rewards by comparing generated answers against $y^\star$. However, RLVR is computationally expensive and the reward signal is sparse, providing same feedback across all tokens regardless of where errors occur. Alternatively, one can train a process reward model (PRM) to provide dense, token-level feedback during RL. However, acquiring labels for PRM training is prohibitively expensive and difficult to scale (Lightman et al., 2023; Zhang et al., 2025). On-policy distillation works (Agarwal et al., 2024; Xu et al., 2024a; Lu & Lab, 2025) address distribution shift by training on the student's own samples, but require a separate, often larger, teacher model to provide supervision. We instead seek a training signal that is *dense*, *on-policy*, and *does not require external teachers or reward models*. This motivates our On-Policy Self-Distillation approach. We summarize the differences of these methods in Table 1.

Algorithm 1 On-Policy Self-Distillation (OPSD)

Require: Reasoning dataset $\mathcal{S} = \{(x_i, y_i^\star)\}_{i=1}^N$; language model p_θ; divergence D (e.g., JSD_β)

1: Let $p_S(\cdot \mid x)$ and $p_T(\cdot \mid x, y^\star)$ be the same model p_θ under different conditioning.
2: **while** not converged **do**
3: Sample a minibatch $\mathcal{B} \subset \mathcal{S}$
4: **for all** $(x, y^\star) \in \mathcal{B}$ **do**
5: Sample on-policy response $\hat{y} \sim p_S(\cdot \mid x)$
6: Compute the token-wise divergence along the student rollout:

$$\ell(x, y^\star) \leftarrow D\big(p_T \,\|\, p_S\big)(\hat{y} \mid x) = \frac{1}{|\hat{y}|} \sum_{n=1}^{|\hat{y}|} D\big(p_T(\cdot \mid \hat{y}_{<n}, x, y^\star) \,\big\|\, p_S(\cdot \mid \hat{y}_{<n}, x)\big)$$

7: Calculate loss $\mathcal{L}_{\mathrm{OPSD}}(\theta) \leftarrow \frac{1}{|\mathcal{B}|} \sum_{(x,y^\star)\in\mathcal{B}} \ell(x, y^\star)$ and update $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}_{\mathrm{OPSD}}(\theta)$

3.2 On-Policy Self-Distillation

Motivation: Learning by understanding solutions. We propose a different perspective inspired by how students learn: when struggling with a problem, rather than extended trial-and-error, a student can examine the solution, understand the reasoning, and internalize the approach. Similarly, if a model has access to the correct answer or reasoning $y^\star$ and is sufficiently capable, it can rationalize the reasoning steps and teach itself—analogous to a student reviewing a solution and retracing why it works. This intuition motivates our framework: we exploit the ground-truth solution $y^\star$ directly as privileged information during training, enabling the model to serve as its own teacher without requiring external reward models or larger teacher models.

Teacher and student policies. We instantiate two conditional distributions from the same language model p_θ by varying the conditioning context. The *teacher policy* conditions on privileged information—both the problem x and the reference solution $y^\star$:

$$p_T(\cdot \mid x, y^\star) \triangleq p_\theta(\cdot \mid x, y^\star).$$

The *student policy* observes only the problem statement, matching the inference-time condition:

$$p_S(\cdot \mid x) \triangleq p_\theta(\cdot \mid x).$$

Both policies share the same parameters θ but differ only in their conditioning context. To encourage the teacher to naturally evaluate the student's generation, we add a prompt asking the teacher to generate a new solution after seeing the reference solution as shown in Figure 2. However, the teacher doesn't generate tokens, it only does rationalization implicitly through prefilling.

On-policy sampling from the student. Given a problem x, the student generates an on-policy response

$$\hat{y} = (\hat{y}_1, \ldots, \hat{y}_{|\hat{y}|}) \sim p_S(\cdot \mid x).$$

Both policies then evaluate this student-generated trajectory. At each position n, they induce *next-token* distributions over $y_n \in \mathcal{V}$ conditioned on the same student prefix:

$$p_S(y_n \mid x, \hat{y}_{<n}), \qquad p_T(y_n \mid x, y^\star, \hat{y}_{<n}),$$

where $\hat{y}_{<n} \triangleq (\hat{y}_1, \ldots, \hat{y}_{n-1})$.

Training objective: Full-vocabulary divergence. We instantiate a *full-vocabulary divergence objective* that matches the teacher and student next-token distributions at each position. Given a student-generated sequence $\hat{y}$, define the trajectory-averaged, token-wise divergence

$$D\big(p_T \,\|\, p_S\big)(\hat{y} \mid x) \triangleq \frac{1}{|\hat{y}|} \sum_{n=1}^{|\hat{y}|} D\Big(p_T(\cdot \mid x, y^\star, \hat{y}_{<n}) \,\Big\|\, p_S(\cdot \mid x, \hat{y}_{<n})\Big), \tag{5}$$

where $p_S(\cdot \mid x, \hat{y}_{<n})$ and $p_T(\cdot \mid x, y^\star, \hat{y}_{<n})$ denote distributions over the next token $y_n \in \mathcal{V}$. Here, D can be any distribution divergence measure such as the *generalized Jensen-Shannon divergence*

JSD_β, defined for a weight $\beta \in [0, 1]$ as:

$$\text{JSD}_\beta(p_T \| p_S) = \beta D_{KL}(p_T \| m) + (1 - \beta) D_{KL}(p_S \| m) \tag{6}$$

where $m = \beta p_T + (1 - \beta) p_S$ is the interpolated mixture distribution. This full-vocabulary formulation provides dense, token-level feedback: the teacher, informed by $y^\star$, exposes the student to the entire distribution over plausible next tokens and guides it toward reasoning paths that lead to the correct answer.

We minimize the expected divergence between teacher and student over on-policy student samples:

$$\mathcal{L}(\theta) = \mathbb{E}_{(x,y^\star)\sim\mathcal{S}} \left[\mathbb{E}_{\hat{y}\sim p_S(\cdot|x)} \left[D\big(p_T \,\|\, p_S\big)(\hat{y} \mid x) \right] \right]. \tag{7}$$

Gradients are backpropagated only through the student policy p_S, while the teacher p_T acts as a fixed full-distribution target conditioned on privileged information $(x, y^\star)$.

Alternative objective: Sampled-token distillation through policy gradient. Following recent on-policy distillation methods (Lu & Lab, 2025), we form a sampled-token reward signal (a reverse-KL signal on sampled actions) and optimize with policy gradient. For each position n in a sampled sequence $\hat{y}$, define the advantage term

$$A_n(x, \hat{y}) = \log p_T(\hat{y}_n \mid x, y^\star, \hat{y}_{<n}) - \log p_S(\hat{y}_n \mid x, \hat{y}_{<n}),$$

and optimize the policy-gradient-style objective

$$\mathcal{L}(\theta) = -\mathbb{E}_{(x,y^\star)\sim\mathcal{S}} \left[\mathbb{E}_{\hat{y}\sim p_S(\cdot|x)} \left[\frac{1}{|\hat{y}|} \sum_{n=1}^{|\hat{y}|} A_n(x, \hat{y}) \times \log p_S(\hat{y}_n \mid x, \hat{y}_{<n}) \right] \right]. \tag{8}$$

$A_n(x, \hat{y})$ is treated as a constant with respect to θ (i.e., gradients do not flow through the advantage), so that gradients take the usual policy-gradient form $A_n \nabla_\theta \log p_S$. Compared to the full-vocabulary divergence objective, this on-policy shaping objective operates only on sampled tokens, using the teacher's log-probabilities to provide dense, trajectory-level shaping signals without explicitly matching the full distribution at each step.

OPSD as dense-reward policy gradient and comparison to STaR. The objective in Equation (8) can be seen as policy gradient with dense, token-level rewards. In Appendix Section D, we formalize this and contrast with STaR (Zelikman et al., 2022), a closely related method that also uses the same model to generate reasoning traces, then performs rejection sampling followed by SFT on correct traces. This procedure can be viewed as policy gradient with a sequence-level binary reward that assigns identical credit to all tokens and vanishes when samples are incorrect. In contrast, OPSD provides feedback at every token position regardless of final-answer correctness.

4 Experiments

We conduct comprehensive experiments to answer the following research questions:

(1) How does OPSD compare to SFT and GRPO in reasoning performance and sample efficiency? (§4.2)
(2) How does OPSD scale across model sizes? (§4.3.1)
(3) What is the effect of generation length on performance and sample efficiency? (§4.3.2)
(4) Does full-vocabulary divergence provide benefits over sampled-token policy gradient? (§4.3.3)

4.1 Experimental Setup

Models and datasets. We experiment with the Qwen3 (Team, 2025b) model family at three scales: Qwen3-1.7B, Qwen3-4B, and Qwen3-8B, using the instruct-tuned versions. For training data, we use the mathematical reasoning subset of OpenThoughts (Guha et al., 2025), sampling up to 30K problem-solution pairs with chain-of-thought reasoning. We evaluate on competition-level mathematics benchmarks including AIME 2024, AIME 2025, HMMT 2025 and Amo-Bench (An et al., 2025b).

Table 2: Performance comparison across mathematical reasoning benchmarks for Qwen3 models from 1.7B to 8B. We report average@16 using suggested sampling parameters from the Qwen3 blog with temperature of 1.2 and generation length of 38k, with detailed parameter in Table 6.

Method	AIME24	AIME25	HMMT25	AMO-Bench	Average
Qwen3-8B					
Base (Instruct)	75.2	68.3	43.1	13.4	50.0
+ SFT	76.3	66.2	44.7	12.9	50.0
+ GRPO	76.7	68.7	45.0	**14.8**	51.3
+ OPSD	**77.5**	**69.8**	**47.1**	14.3	**52.2**
Qwen3-4B					
Base (Instruct)	74.6	65.8	40.3	12.4	48.3
+ SFT	75.2	66.3	44.4	12.5	49.6
+ GRPO	75.6	**67.1**	42.7	12.8	49.6
+ OPSD	**76.0**	66.9	**45.8**	**13.5**	**50.6**
Qwen3-1.7B					
Base (Instruct)	50.2	35.2	25.4	4.3	28.8
+ SFT	48.3	36.3	23.3	3.9	28.0
+ GRPO	**52.1**	38.3	**26.7**	4.5	**30.5**
+ OPSD	51.4	**39.5**	25.8	**5.0**	30.4

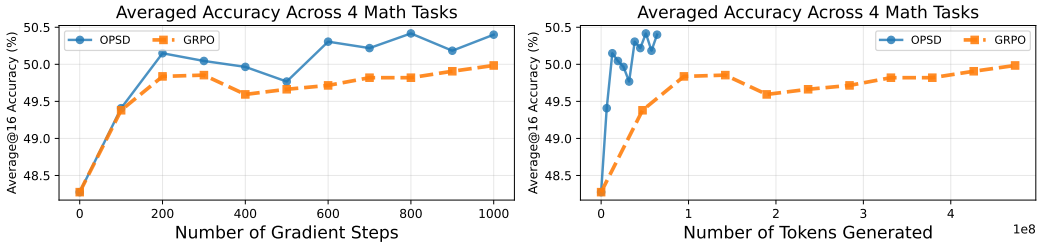

Figure 3: **Token Efficiency of OPSD.** We compare OPSD and GRPO on Qwen3-4B under the same effective training batch size, reporting average@16 performance as a function of gradient update steps and total generated tokens. Both methods are trained with the same effective batch size in terms of sampled generations per update, but differ in generation length: each generation is capped at 2048 tokens for OPSD and 16384 tokens for GRPO. OPSD achieves comparable or better performance with substantially fewer generated tokens, resulting in lower sampling cost and reduced training time. In this experiment, OPSD can be 4-8$\times$ more token-efficient than GRPO.

Baselines. We compare against two methods trained on the same dataset: (1) **SFT**, standard supervised fine-tuning on expert trajectories, which can be seen as off-policy distillation from a more powerful LLM that generated the reasoning traces; (2) **GRPO** (Shao et al., 2024), group relative policy optimization with binary outcome rewards verified against ground-truth answers. The max generation length is set to 16k.

Implementation details. For GRPO, we sample 8 responses per problem. For OPSD, we sample 1 response per problem. We use Adam optimizer with a learning rate of 1e-5, warmup ratio of 0.1, and cosine learning rate decay. For the divergence measure in Eq. 5, we use $\text{JSD}_{\beta=0.5}$. Importantly, we fix the teacher policy to be the initial policy, rather than the currently updating learning policy, as we find this helps stabilize training and implicitly acts as regularization to prevent excessive deviation from the initial policy. All experiments are conducted on 8×A100 GPUs with LoRA (Hu et al., 2022). More experimental details are in Appendix C.

4.2 Main Results

Table 2 reports results on competition-level mathematical reasoning benchmarks. OPSD consistently outperforms SFT and improves over the base model across scales; it matches or exceeds GRPO at 4B/8B, and is comparable at 1.7B. Notably, OPSD accomplishes these gains using only a single rollout per problem, whereas GRPO requires 8 rollouts, demonstrating improved sample efficiency.

Superior Token Efficiency from Dense Teacher Feedback. In addition to improved accuracy, OPSD is significantly more token-efficient than GRPO. Figure 3 compares the two methods under the same effective training batch size on Qwen3-4B. While GRPO relies on 8 rollouts with long generation budgets of 16k, OPSD achieves higher performance using substantially fewer generated tokens of 2k and needs only 1 rollout per prompt. This efficiency stems from dense token-level supervision from the teacher distribution, reducing sampling cost and training time without sacrificing performance. We hypothesize that the early tokens are more important for distillation than the later tokens, as the earlier tokens can represent more important branching points.

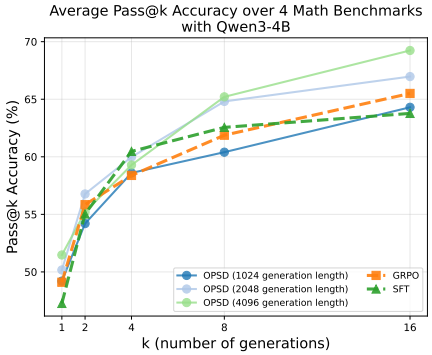

Figure 4: Pass@K performance averaged across four mathematical reasoning benchmarks for Qwen3-4B. We study the effect of the generation length of on-policy sampled student responses in OPSD, comparing 1024, 2048, and 4096 tokens. Longer generations provide more teacher signals. Increasing the generation length from 1k to 2k and 4k consistently improves pass@K, with both 2k and 4k substantially outperforming the 1k setting.

4.3 Discussions

4.3.1 Effect of Model Scale

Our method relies on the teacher policy's ability to rationalize reference solutions when conditioned on privileged information. Under a fixed dataset, this capability depends on sufficient model capacity and is expected to scale with model size. We therefore hypothesize that OPSD becomes increasingly effective as models grow more capable of leveraging privileged context. To evaluate this, we apply OPSD to the Qwen3 family at three scales: 1.7B, 4B, and 8B parameters. As shown in Table 2, OPSD provides limited gains over GRPO at the 1.7B scale although OPSD still improves over base and SFT at 1.7B., while yielding progressively larger improvements at the 4B and 8B scales, consistent with our hypothesis.

4.3.2 Effect of Generation Length

Since our objective operates at the token level (Eq. 5), the number of generated tokens per sample directly determines the amount of supervision signal available to the student. Longer sequences expose the student to more teacher feedback, but they also increase computational cost and may introduce noisy or uninformative continuations.

To study this trade-off, we conduct an ablation on Qwen3-4B by varying the generation length of on-policy sampled student responses among 1024, 2048, and 4096 tokens and use full-vocabulary logit distillation. As shown in Figure 4, increasing the generation length leads to clear improvements in pass@K performance. In particular, both the 2048-token and 4096-token settings significantly outperform the 1024-token baseline, indicating that longer generations provide more effective reasoning supervision.

Table 3: Ablation on divergence computation strategies for OPSD on Qwen3-4B with 2048 generation length for distillation. We report pass@8 accuracy on AIME25 and HMMT25. Full-distribution objectives (logit distillation) outperform sampled-token objectives.

Method Variant	AIME25	HMMT25
OPSD w/ Full-vocabulary logit distillation (Agarwal et al., 2024)	**84.1**	**60.0**
OPSD w/ Sampled-token distillation (Lu & Lab, 2025)	82.1	57.3

4.3.3 Learning Objective Comparison: Full Vocabulary Logits Distillation vs. Sampled-Token Distillation

Our objective in Eq. 5 is defined as a per-token discrepancy between the teacher and student *distributions*. In practice, OPSD can instantiate this objective in two ways. (1) **Full-vocabulary logit distillation** (as in GKD (Agarwal et al., 2024)): for each token position, we compute $D(p_T \,\|\, p_S)$ over the entire vocabulary via a full softmax, yielding a proper token-level f-divergence between the two policies. (2) **Sampled-token advantage policy-gradient objective** (as in the on-policy distillation method of Lu & Lab (2025)): we evaluate teacher and student log-probabilities only at the token actually sampled by the student, $\hat{y}_n$, and use the reverse-KL term as a scalar advantage inside a policy-gradient-style loss. Thus, the first variant directly matches full token distributions, whereas the second optimizes an on-policy RL objective shaped by the teacher's log-probabilities rather than a full-distribution divergence. We compare these variants on Qwen3-4B using a 2048-token generation budget during distillation. Table 3 summarizes the results. The full-vocabulary divergence objective provides a consistent gain over the sampled-token objective, improving AIME25 from 82.1% to 84.1% and HMMT25 from 57.3% to 60.0%. This suggests that exposing the student to the full teacher distribution offers richer supervision than relying solely on per-token on-policy shaping. However, the full-vocabulary computation incurs higher peak memory usage due to storing vocabulary-sized logits at every position, indicating a trade-off between performance and efficiency.

5 Related Work

LLM Self-Training. A growing body of work shows that LLMs can improve by generating and leveraging their own supervision signals (Allen-Zhu & Li, 2020; Xu et al., 2024b; Chen et al., 2024; Wang et al., 2023; Sun et al., 2023; Yuan et al., 2024; Yang et al., 2024). Closest to our approach is *context distillation* (Snell et al., 2022), where a model acts as both teacher and student by conditioning the teacher on privileged context and performing off-policy, hard distillation via SFT on generated outputs. In reasoning, ReST (Gulcehre et al., 2023) and STaR (Zelikman et al., 2022) similarly rely on iterative self-training loops that generate, filter, and fine-tune on successful reasoning trajectories. More recently, in-context editing (Qi et al., 2025) demonstrates that context-induced knowledge can be internalized through soft distillation by directly minimizing divergences between distributions. OPSD differs from prior self-training methods by performing *on-policy*, *soft* distillation on the student's own rollouts for reasoning tasks, using per-token distribution matching rather than SFT on generated rationales. **On-Policy Distillation.** On-policy distillation methods train a student directly on trajectories sampled from its own policy while a teacher provides per-token guidance via KL-based regularization or related objectives (Agarwal et al., 2024; Xu et al., 2024a; Gu et al., 2024; Lu & Lab, 2025; Xiaomi, 2026; Yang et al., 2025). By optimizing on the student's visitation distribution, these methods mitigate distribution shift, but typically rely on a distinct and often larger teacher model. In contrast, we study whether an LLM can teach itself by conditioning on privileged answer information and using its own reasoning capability to guide a weaker version of itself. More broadly, on-policy supervision has long been studied in robotics and RL, notably in DAgger (Ross et al., 2011), where expert feedback is provided on states visited by the learner. We discuss more related works in Appendix B.

6 Conclusion

We introduced On-Policy Self-Distillation (OPSD), a simple yet effective framework for post-training large language models on reasoning tasks. The intuition behind OPSD is that a sufficiently capable reasoning LLM can teach itself when it has access to privileged information about the answer to a reasoning problem, utilizing its own rationalization ability to grade its weaker self without access to the ground truth. We experimentally demonstrated that OPSD achieves better performance than off-policy distillation/SFT, and performs on par with or better than GRPO, while exhibiting significantly better sample efficiency than GRPO. Our ablation studies reveal that sufficiently large language models are required for successful self-distillation, and that generating more tokens during the online sampling phase and full-vocabulary logit distillation leads to improved learning.

REFERENCES

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

Zeyuan Allen-Zhu and Yuanzhi Li. Towards understanding ensemble, knowledge distillation and self-distillation in deep learning. In *The Eleventh International Conference on Learning Representations*, 2020.

Chenxin An, Zhihui Xie, Xiaonan Li, Lei Li, Jun Zhang, Shansan Gong, Ming Zhong, Jingjing Xu, Xipeng Qiu, Mingxuan Wang, and Lingpeng Kong. Polaris: A post-training recipe for scaling reinforcement learning on advanced reasoning models, 2025a. URL https://hkunlp.github.io/blog/2025/Polaris.

Shengnan An, Xunliang Cai, Xuezhi Cao, Xiaoyu Li, Yehao Lin, Junlin Liu, Xinxuan Lv, Dan Ma, Xuanlin Wang, Ziwen Wang, et al. Amo-bench: Large language models still struggle in high school math competitions. *arXiv preprint arXiv:2510.26768*, 2025b.

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

Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V Le, Sergey Levine, and Yi Ma. Sft memorizes, rl generalizes: A comparative study of foundation model post-training. *arXiv preprint arXiv:2501.17161*, 2025.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Minillm: Knowledge distillation of large language models. In *ICLR*, 2024.

Etash Guha, Ryan Marten, Sedrick Keh, Negin Raoof, Georgios Smyrnis, Hritik Bansal, Marianna Nezhurina, Jean Mercat, Trung Vu, Zayne Sprague, Ashima Suvarna, Benjamin Feuer, Liangyu Chen, Zaid Khan, Eric Frankel, Sachin Grover, Caroline Choi, Niklas Muennighoff, Shiye Su, Wanjia Zhao, John Yang, Shreyas Pimpalgaonkar, Kartik Sharma, Charlie Cheng-Jie Ji, Yichuan Deng, Sarah Pratt, Vivek Ramanujan, Jon Saad-Falcon, Jeffrey Li, Achal Dave, Alon Albalak, Kushal Arora, Blake Wulfe, Chinmay Hegde, Greg Durrett, Sewoong Oh, Mohit Bansal, Saadia Gabriel, Aditya Grover, Kai-Wei Chang, Vaishaal Shankar, Aaron Gokaslan, Mike A. Merrill, Tatsunori Hashimoto, Yejin Choi, Jenia Jitsev, Reinhard Heckel, Maheswaran Sathiamoorthy, Alexandros G. Dimakis, and Ludwig Schmidt. Openthoughts: Data recipes for reasoning models, 2025. URL https://arxiv.org/abs/2506.04178.

Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, et al. Reinforced self-training (rest) for language modeling. *arXiv preprint arXiv:2308.08998*, 2023.

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

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015. URL https://arxiv.org/abs/1503.02531.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Maggie Huan, Yuetai Li, Tuney Zheng, Xiaoyu Xu, Seungone Kim, Minxin Du, Radha Poovendran, Graham Neubig, and Xiang Yue. Does math reasoning improve general llm capabilities? understanding transferability of llm reasoning. *arXiv preprint arXiv:2507.00432*, 2025.

Yoon Kim and Alexander M Rush. Sequence-level knowledge distillation. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pp. 1317–1327, 2016.

Jia LI, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Costa Huang, Kashif Rasul, Longhui Yu, Albert Jiang, Ziju Shen, Zihan Qin, Bin Dong, Li Zhou, Yann Fleureau, Guillaume Lample, and Stanislas Polu. Numinamath. https://github.com/project-numina/aimo-progress-prize/blob/main/report/numina_dataset.pdf, 2024.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2023.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Kevin Lu and Thinking Machines Lab. On-policy distillation. *Thinking Machines Lab: Connectionism*, 2025. doi: 10.64434/tml.20251026. https://thinkingmachines.ai/blog/on-policy-distillation.

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

Moni Naor. Evaluation may be easier than generation. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pp. 74–83, 1996.

Keiran Paster, Marco Dos Santos, Zhangir Azerbayev, and Jimmy Ba. Openwebmath: An open dataset of high-quality mathematical web text, 2023.

Siyuan Qi, Bangcheng Yang, Kailin Jiang, Xiaobo Wang, Jiaqi Li, Yifan Zhong, Yaodong Yang, and Zilong Zheng. In-context editing: Learning knowledge from self-induced distributions. In *The Thirteenth International Conference on Learning Representations*, 2025.

Abhinav Rastogi, Albert Q Jiang, Andy Lo, Gabrielle Berrada, Guillaume Lample, Jason Rute, Joep Barmentlo, Karmesh Yadav, Kartik Khandelwal, Khyathi Raghavi Chandu, et al. Magistral. *arXiv preprint arXiv:2506.10910*, 2025.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 627–635. JMLR Workshop and Conference Proceedings, 2011.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

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

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.

Charlie Snell, Dan Klein, and Ruiqi Zhong. Learning by distilling context. *arXiv preprint arXiv:2209.15189*, 2022.

Zhiqing Sun, Yikang Shen, Qinhong Zhou, Hongxin Zhang, Zhenfang Chen, David Cox, Yiming Yang, and Chuang Gan. Principle-driven self-alignment of language models from scratch with minimal human supervision. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=p40XRfBX96.

Zhiqing Sun, Longhui Yu, Yikang Shen, Weiyang Liu, Yiming Yang, Sean Welleck, and Chuang Gan. Easy-to-hard generalization: Scalable alignment beyond human supervision. *Advances in Neural Information Processing Systems*, 37:51118–51168, 2024.

Kimi Team, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, et al. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*, 2025.

OpenThoughts Team. Open Thoughts. https://open-thoughts.ai, January 2025a.

Qwen Team. Qwen3 technical report, 2025b. URL https://arxiv.org/abs/2505.09388.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, pp. 13484–13508, 2023.

LLM-Core Xiaomi. Mimo-v2-flash technical report, 2026. URL https://arxiv.org/abs/2601.02780.

Wenda Xu, Rujun Han, Zifeng Wang, Long Le, Dhruv Madeka, Lei Li, William Yang Wang, Rishabh Agarwal, Chen-Yu Lee, and Tomas Pfister. Speculative knowledge distillation: Bridging the teacher-student gap through interleaved sampling. In *The Thirteenth International Conference on Learning Representations*, 2024a.

Xiaohan Xu, Ming Li, Chongyang Tao, Tao Shen, Reynold Cheng, Jinyang Li, Can Xu, Dacheng Tao, and Tianyi Zhou. A survey on knowledge distillation of large language models. *CoRR*, 2024b.

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

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

Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. Limo: Less is more for reasoning, 2025. URL https://arxiv.org/abs/2502.03387.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*, 2023.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, et al. Dapo: An open-source llm reinforcement learning system at scale, 2025. *URL https://arxiv. org/abs/2503.14476*, 2025.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason E Weston. Self-rewarding language models. In *International Conference on Machine Learning*, pp. 57905–57923. PMLR, 2024.

Yu Yue, Yufeng Yuan, Qiying Yu, Xiaochen Zuo, Ruofei Zhu, Wenyuan Xu, Jiaze Chen, Chengyi Wang, TianTian Fan, Zhengyin Du, et al. Vapo: Efficient and reliable reinforcement learning for advanced reasoning tasks. *arXiv preprint arXiv:2504.05118*, 2025.

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

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

Siyan Zhao, Mengchen Liu, Jing Huang, Miao Liu, Chenyu Wang, Bo Liu, Yuandong Tian, Guan Pang, Sean Bell, Aditya Grover, et al. Inpainting-guided policy optimization for diffusion large language models. *arXiv preprint arXiv:2509.10396*, 2025.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, et al. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*, 2025.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. Lima: less is more for alignment. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, pp. 55006–55021, 2023.

A Limitations and Future Directions

Due to computational constraints, our experiments are limited to models up to 8B parameters. While we observe that larger models benefit more from OPSD—consistent with our hypothesis that self-rationalization requires sufficient model capacity—it remains an open question whether this trend continues at scales beyond 8B parameters, such as 70B or larger frontier models. Several promising directions warrant further investigation. First, our current framework does not explicitly leverage correctness verification of generated answers; incorporating such signals could provide additional learning objectives beyond distribution matching. Finally, problem difficulty plays a crucial role in self-distillation: if reasoning problems exceed the model's comprehension threshold, the teacher policy cannot provide meaningful supervision even with access to ground-truth solutions. This suggests that curriculum learning strategies—gradually increasing problem difficulty as the model improves—could enhance training effectiveness. Exploring adaptive curricula that maintain problems at the frontier of model capabilities represents an important direction for scaling OPSD to more challenging reasoning tasks.

B Related Work

LLM Self-Training. Our work connects to a line of research showing that LLMs can improve by generating and exploiting their own supervision signals (Allen-Zhu & Li, 2020; Xu et al., 2024b; Chen et al., 2024; Wang et al., 2023; Sun et al., 2023; Yuan et al., 2024; Yang et al., 2024). Closest in spirit is *context distillation* (Snell et al., 2022), which uses the same underlying model as both teacher and student by providing the teacher with privileged context and then SFT the student on the teacher's *generated* outputs without context. This can be viewed as off-policy, hard distillation, where the learning signal is a discrete token sequence. In the reasoning domain, ReST (Gulcehre et al., 2023) and STaR (Zelikman et al., 2022) similarly rely on iterative self-training loops—generate rationales, filter by rewards or ground-truth answers, and fine-tune on successful trajectories—again yielding hard distillation. More recently, in-context editing (Qi et al., 2025) shows that *context-induced* knowledge can be internalized via soft distillation by directly minimizing divergences between distributions, and demonstrates this effect in knowledge editing settings. OPSD differs from these approaches in that we perform *on-policy*, soft distillation on the student's own rollouts for reasoning tasks: the teacher's supervision is per-token distribution matching rather than generating a rationale for SFT. OPSD frames reasoning improvement as learning a conditional distribution induced jointly by the dataset's ground-truth solutions and the model's own reasoning ability.

On-Policy Distillation methods train a student model directly on trajectories sampled from its own policy, while a teacher model provides per-token guidance through KL-based regularization or related objectives (Agarwal et al., 2024; Xu et al., 2024a; Gu et al., 2024; Lu & Lab, 2025; Xiaomi, 2026; Yang et al., 2025). These approaches mitigate distribution shift by optimizing directly on the student's visitation distribution, but they typically rely on a distinct and often larger teacher model. In this work, we explore whether an LLM can teach itself by conditioning on more privileged answer information and leveraging its own reasoning capability to guide a weaker version of itself toward improved reasoning. On-policy training paradigms are also widely used in robotics and deep reinforcement learning, such as DAgger (Ross et al., 2011), where a human teacher provides corrective supervision on the states visited by the student policy.

Improving LLM Reasoning through SFT and RL. SFT and RL are two primary methods for improving LLM reasoning ability. SFT on high-quality reasoning traces has demonstrated strong performance (Yu et al., 2023; LI et al., 2024; Paster et al., 2023; Team, 2025a; Ye et al., 2025; Muennighoff et al., 2025; Zhou et al., 2023). However, prior work shows that SFT can rely on memorization rather than robust generalization (Chu et al., 2025). In contrast, RL optimizes directly for outcome-based objectives can exhibit better generalization (Huan et al., 2025). More recent algorithms such as GRPO (Guo et al., 2025; Shao et al., 2024) enable scalable RL by estimating advantages from group-level rewards without requiring an explicit critic as in PPO (Schulman et al., 2017). Building on this line of work, a growing body of research highlights the effectiveness of RLVR for reasoning tasks (Yu et al., 2025; Liu et al., 2025; Yue et al., 2025; An et al., 2025a; Zheng et al., 2025).

C EXPERIMENTAL DETAILS

We provide the training and evaluation configurations for our SFT, GRPO and OPSD experiments in Tables 5, 4 and 6. Both GRPO and OPSD methods use the same base hyperparameters where applicable to ensure fair comparison.

Table 4: Training Configuration for GRPO and OPSD

Parameter	**GRPO**	**OPSD**
Learning Rate	2×10^{-5}	2×10^{-5}
Batch Size (per device)	1	1
Gradient Accumulation Steps	4	4
Effective Batch Size	32	32
LoRA Rank (r)	64	64
LoRA Alpha (α)	128	128
LoRA Target Modules	q_proj, k_proj, v_proj, o_proj, gate_proj, up_proj, down_proj	
Max Completion Length	16000	2048
Number of Generations per Prompt	8	1
Temperature	1.2	1.2
KL Coefficient (β)	0.0	–

Table 5: Training Configuration for SFT.

Parameter	**SFT**
Learning Rate	2×10^{-5}
Batch Size (per device)	2
Gradient Accumulation Steps	4
Effective Batch Size	64
LoRA Rank (r)	64
LoRA Alpha (α)	128
LoRA Target Modules	q_proj, k_proj, v_proj, o_proj, gate_proj, up_proj, down_proj
Max Sequence Length	16000
Number of Training Epochs	4
Training Dataset Size	30k

Table 6: Evaluation Parameters.

Parameter	**Value**
Max New Tokens	38912
Thinking Mode	Enabled
Top-p	0.95
Top-k	-1
Min-p	0.0
Presence Penalty	0.0
Samples per Prompt	16

All experiments were conducted using 8 A100 GPUs with gradient checkpointing and Flash Attention 2 for memory efficiency. We use the AdamW (Loshchilov & Hutter, 2017) optimizer and bfloat16 precision for all training runs. For OPSD, unless otherwise stated, we used full-vocabulary logit distillation.

D Policy-Gradient Interpretation of OPSD and Comparison to STaR

Our OPSD objective in Equation (8) can be interpreted as a policy-gradient update with a *dense, token-level* reward signal derived from privileged information. In this section, we show: (1) OPSD can be seen as a dense-reward policy gradient, and (2) we contrast OPSD with STaR, demonstrating that STaR's learning signal is *sequence-level* while OPSD is *token-level*.

D.1 STaR as Sequence-Level Policy-Gradient

STaR (Zelikman et al., 2022) can be viewed as an approximation to an RL-style policy gradient objective. The language model p_θ induces a joint distribution over rationale r and answer y:

$$p_\theta(r, y \mid x) = p_\theta(r \mid x)\, p_\theta(y \mid x, r),$$

where the model first samples a latent rationale r before predicting the final answer y. Given an indicator reward $R(y) = \mathbf{1}(y = y^\star)$, the expected return across the dataset $\mathcal{S} = \{(x_i, y_i^\star)\}_{i=1}^N$ is

$$J_{\text{STaR}}(\theta) = \sum_{i=1}^{N} \mathbb{E}_{(r,y)\sim p_\theta(\cdot|x_i)} \big[\mathbf{1}(y = y_i^\star)\big]. \tag{9}$$

Applying the log-derivative trick yields a policy gradient:

$$\nabla_\theta J_{\text{STaR}}(\theta) = \sum_{i=1}^{N} \mathbb{E}_{(r,y)\sim p_\theta(\cdot|x_i)} \Big[\mathbf{1}(y = y_i^\star)\, \nabla_\theta \log p_\theta(r, y \mid x_i)\Big]. \tag{10}$$

Note that the indicator function discards the gradient for all sampled rationales that do not lead to the correct answer $y_i^\star$: this corresponds to the filtering step in STaR.

One limitation is that STaR's reward is *sequence-level*: the binary indicator $\mathbf{1}(y = y^\star)$ provides the same signal to all tokens in a trajectory, offering no intermediate credit assignment. When all sampled trajectories are all incorrect, the learning signal vanishes.

D.2 OPSD as Dense-Reward Policy Gradient

The sampled-token objective in Equation (8) can also be viewed as a policy-gradient method, but with a token-level reward. Fix a training pair $(x, y^\star)$ and let the student generate a trajectory $\hat{y} \sim p_S(\cdot \mid x)$. At each position n, define the per-token reward:

$$r_n(x, \hat{y}) \triangleq \log p_T(\hat{y}_n \mid x, y^\star, \hat{y}_{<n}) - \log p_S(\hat{y}_n \mid x, \hat{y}_{<n}).$$

This reward measures how much the privileged teacher prefers the sampled token $\hat{y}_n$ relative to the student. As stated in the main text, we treat r_n (equivalently, the advantage A_n) as a constant with respect to θ when computing gradients—that is, we stop gradients through both p_T and p_S in the reward computation. Under this treatment, the gradient of Equation (8) takes the standard policy-gradient form:

$$\nabla_\theta \mathcal{L}(\theta) = -\mathbb{E}_{(x,y^\star)\sim\mathcal{S}} \left[\mathbb{E}_{\hat{y}\sim p_S(\cdot|x)} \left[\frac{1}{|\hat{y}|} \sum_{n=1}^{|\hat{y}|} r_n(x, \hat{y})\, \nabla_\theta \log p_S(\hat{y}_n \mid x, \hat{y}_{<n}) \right] \right],$$

which corresponds to maximizing the expected per-token reward along on-policy student rollouts:

$$J_{OPSD}(\theta) = \mathbb{E}_{(x,y^\star)\sim\mathcal{S}} \left[\mathbb{E}_{\hat{y}\sim p_S(\cdot|x)} \left[\frac{1}{|\hat{y}|} \sum_{n=1}^{|\hat{y}|} r_n(x, \hat{y}) \right] \right].$$

This reward is dense: it provides a learning signal at every token position, regardless of whether the final answer is correct.

Comparison. Both STaR and OPSD can be understood as policy-gradient methods, but their reward structures differ fundamentally. STaR uses a sequence-level indicator $\mathbf{1}(y = y^\star)$ that assigns the same signal to all tokens; when all sampled trajectories are incorrect, the learning signal vanishes entirely. In contrast, OPSD provides a token-level reward r_n at every position, enabling fine-grained credit assignment even when the final answer is wrong.