LOOPDISTILL: STUDENT-AWARE KNOWLEDGE DISTILLATION FOR LOOPED LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Looped language models (LoopLMs) increase effective depth by repeatedly applying shared Transformer blocks, offering a parameter-efficient approach to reasoning. Knowledge distillation typically transfers reasoning by training a smaller model on chain-of-thought trajectories from a stronger teacher. For a LoopLM, however, readouts at different recurrent depths can generate reasoning trajectories with different lengths and structures. A single teacher trajectory may therefore provide uneven supervision across loops, while training only the final readout leaves intermediate loops without direct reasoning targets. We introduce LoopDistill, a student-aware distillation framework that uses trajectories generated at selected depths to construct loop-specific teacher supervision. Each selected loop is trained on a target adapted to its own reasoning behavior. Using Ouro-2.6B-Base and two teachers, we evaluate LoopDistill on AIME24, AIME25, AIME26, and OlympiadBench. With Qwen3-30B-A3B as the teacher, LoopDistill outperforms the competing baseline on every benchmark, by up to 18.9 percentage points.

1 INTRODUCTION

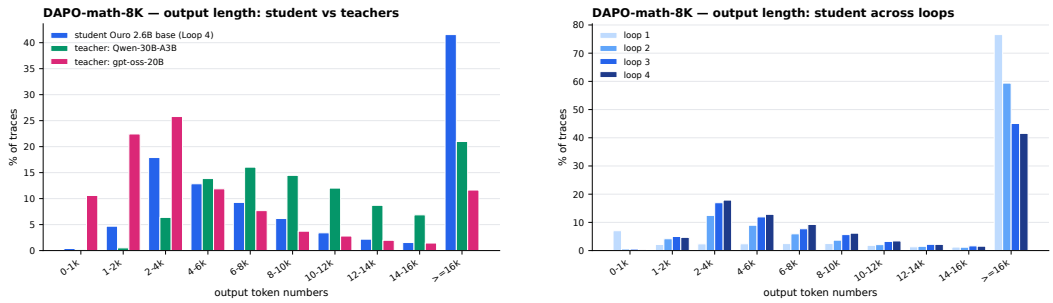
Recent advances in language model reasoning have increasingly explored *latent computation* as an alternative to scaling reasoning through explicit chain-of-thought (CoT) generation. Among these approaches, *looped language models* (LoopLMs) repeatedly apply shared Transformer blocks to update hidden representations during token prediction. This recurrent computation increases effective depth without a proportional increase in parameter count (Saunshi et al., 2025; Zhu et al., 2025). Recent LoopLMs have achieved reasoning performance comparable to that of conventional LLMs with substantially more parameters (Zhu et al., 2025; Yang et al., 2026a), highlighting recurrent depth as a promising axis for scaling language model reasoning.

At the same time, *knowledge distillation* has become an important paradigm for transferring the reasoning capabilities of large language models (LLMs) to smaller students. In reasoning distillation, a teacher typically generates step-by-step chain-of-thought (CoT) trajectories that supervise the student (Magister et al., 2023; Shridhar et al., 2023). Yet these trajectories are drawn from the teacher's distribution: during training, the student learns to continue teacher-generated prefixes, whereas at inference it must continue its own outputs. This distribution mismatch has motivated methods that incorporate student-generated sequences into distillation (Agarwal et al., 2024a; Ko et al., 2024). In reasoning tasks, the usefulness of a teacher trajectory likewise depends on its alignment with the student's current capabilities and generation behavior (Li et al., 2025; Yang et al., 2026b). Together, these findings motivate adapting reasoning supervision to the student's own trajectories.

This concern is especially relevant for LoopLMs. A conventional teacher generates its reasoning through explicit tokens, whereas a LoopLM repeatedly applies shared parameters and can produce different trajectories at different recurrent depths (Zhu et al., 2025; Popescu et al., 2026). On DAPO-Math-8k-Stratified (Yu et al., 2025), the student's output-length distribution differs from those of both teachers (Figure 1a) and changes across readout depths (Figure 1b). The teachers also differ: gpt-oss-20b(OpenAI et al., 2025) places substantial mass on trajectories under 4k tokens and on those reaching the 16k-token generation limit, whereas Qwen3-30B-A3B(Yang et al., 2025) has more trajectories at intermediate lengths (Figure 1a).

Consistent with the concern that teacher trajectories may be unsuitable

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.



(a) Student and teacher trajectories.

(b) Student trajectories at different recurrent depths.

Figure 1: Generation-length distributions on DAPO-Math-8k-Stratified. **(a)** compares trajectories generated by Ouro-2.6B-Base and the teacher models. **(b)** compares student trajectories generated with a fixed exit depth at each loop. Length varies both between teachers and the student and across the student's recurrent depths, motivating loop-specific rewriting.

for the student, direct distillation from gpt-oss-20b reduces accuracy below the Ouro-2.6B-Base model on all four mathematical benchmarks (Table 1). These observations motivate *student-aware distillation*: constructing teacher supervision that remains correct while better matching the reasoning trajectories the student naturally generates.

How to effectively distill reasoning into LoopLMs remains comparatively underexplored. Existing studies have begun to adapt training signals to recurrent computation. For example, recursive Transformers have used knowledge distillation to preserve the behavior of an original non-recurrent model (Bae et al., 2025); RELAY aligns individual CoT reasoning steps with recurrent iterations and introduces iteration-wise supervision (Yu et al., 2026); and recent latent-reasoning methods such as LOTUS directly supervise intermediate latent computation with explicit reasoning targets (Fan et al., 2026). Relatedly, RLTT shows that standard post-training objectives can be suboptimal for LoopLMs when they assign learning signals only to the final recurrent state, motivating supervision that explicitly respects recurrent computation (Williams & Tureci, 2026). However, these approaches do not address a complementary challenge that arises when *distilling a stronger teacher LLM into a pretrained LoopLM*: the teacher and student can exhibit substantially different reasoning distributions, and this mismatch itself can vary across the student's recurrent loops.

In this work, we study *student-aware reasoning distillation for looped language models*. Our key observation is that a LoopLM should not necessarily be trained to reproduce the same teacher reasoning trajectory at every recurrent depth. Instead, supervision should account for both the reasoning behavior naturally accessible to the student at each loop and the interactions among the resulting loop-specific learning objectives. To this end, we introduce LoopDistill, a distillation framework that explicitly aligns teacher supervision with the recurrent computation of a LoopLM.

At the core of LoopDistill is an **trajectory adaptation**. A teacher's reasoning trajectory is not necessarily a suitable distillation target for a looped student: teachers differ in how they structure their solutions, while the student's reasoning behavior also changes across recurrent depths. For each loop, we decode a student trajectory and use it to guide the teacher in constructing a corrected, student-compatible solution. The harness accommodates different rewriting strategies according to the teacher's output pattern, rather than imposing a single transformation on every teacher trajectory. It then verifies each candidate for answer correctness, length, and similarity to the student's CoT format, returning feedback for further revision when a candidate fails. In this way, LoopDistill converts diverse teacher solutions into verified supervision targets tailored to the student's computation at each loop.

We further introduce **multi-loop supervision** to make use of these depth-specific targets. Conventional distillation applies its reasoning loss to the final output, leaving the student's intermediate recurrent computations without direct trajectory supervision. Instead, we decode the output at each supervised loop and train it against the rewrite constructed for that depth. This gives intermediate loops explicit learning signals suited to the reasoning they can produce, while the final loop learns

from its own verified target. The resulting training objective aligns supervision with the student's recurrent computation rather than asking every loop to imitate the same teacher trajectory.

We evaluate LoopDistill using Ouro-2.6B-Base as the looped student and Qwen3-30B-A3B and gpt-oss-20b as teachers on four mathematical reasoning benchmarks. Compared with direct distillation of teacher-generated trajectories (KD), LoopDistill improves accuracy in all four benchmarks, with absolute gains up to 18.9 percentage points.

In summary, our main contributions are as follows:

- We introduce LoopDistill, which constructs a verified reasoning target for each selected recurrent depth. A teacher rewrites the corresponding student rollout using its original solution as guidance; accepted rewrites satisfy correctness, length, and format checks and supervise their respective student readouts.
- We evaluate LoopDistill with Ouro-2.6B-Base, two teachers, and four mathematical reasoning benchmarks. It outperforms direct knowledge distillation in all eight teacher–benchmark settings, with accuracy gains of up to 18.9 percentage points.

2 Related Work

2.1 Looped Language Models

Looped, or recurrent-depth, Transformers increase effective computation by repeatedly applying shared Transformer blocks, providing a parameter-efficient alternative to scaling model depth. Recent work has revisited this paradigm for modern language models. Relaxed Recursive Transformers convert pretrained Transformers into compact recursive models through cross-layer parameter sharing and loop-specific low-rank adaptation (Bae et al., 2025). More recently, Ouro develops LoopLMs directly at pretraining scale and shows that recurrent latent computation can improve parameter efficiency and reasoning capability (Zhu et al., 2025).

A growing line of work investigates how reasoning supervision interacts with recurrent computation. RELAY aligns explicit CoT steps with loop iterations and applies iteration-wise supervision (Yu et al., 2026), while LOTUS supervises latent representations using corresponding CoT steps (Fan et al., 2026). Reinforcement learning has also been applied to recurrent models: RLTT distributes reward across Ouro's latent thought trajectory (Williams & Tureci, 2026); and Loopie applies sequence-level reinforcement learning to looped mixture-of-experts models on mathematical and coding tasks after supervised training (Gao et al., 2026).

2.2 Knowledge Distillation for LLM Reasoning

Knowledge distillation transfers capabilities from a large teacher to a smaller student. Sequence-level distillation trains on teacher-generated sequences (Kim & Rush, 2016), while reasoning distillation uses teacher-generated chain-of-thought trajectories (Magister et al., 2023). These methods have transferred reasoning abilities from substantially larger models to smaller students.

Recent work examines which trajectories and objectives provide effective supervision. MiniLLM uses reverse KL to better match generative language-model distributions (Gu et al., 2024). Generalized Knowledge Distillation (GKD) reduces the train–inference mismatch by sampling student trajectories and obtaining token-level teacher feedback on their prefixes (Agarwal et al., 2024b). DistiLLM likewise incorporates student-generated outputs through an adaptive distillation objective (Ko et al., 2024). However, a student rollout can enter an incorrect reasoning path from which continuing to the correct solution is difficult. This *prefix failure* can make teacher feedback on subsequent student-generated prefixes less useful. Trajectory-Refined Distillation (TRD) addresses this issue by having the teacher revise the student's rollout before distillation (Jiang et al., 2026).

Our work builds on trajectory refinement for a different architectural setting. A LoopLM can produce different rollouts when decoded at different recurrent depths, so a refined target constructed for one readout need not suit another. We sample rollouts from individual student loops, construct and verify a teacher rewrite for each, and supervise the corresponding loop with its own target. Thus, the target construction and the location of supervision both account for recurrent depth.

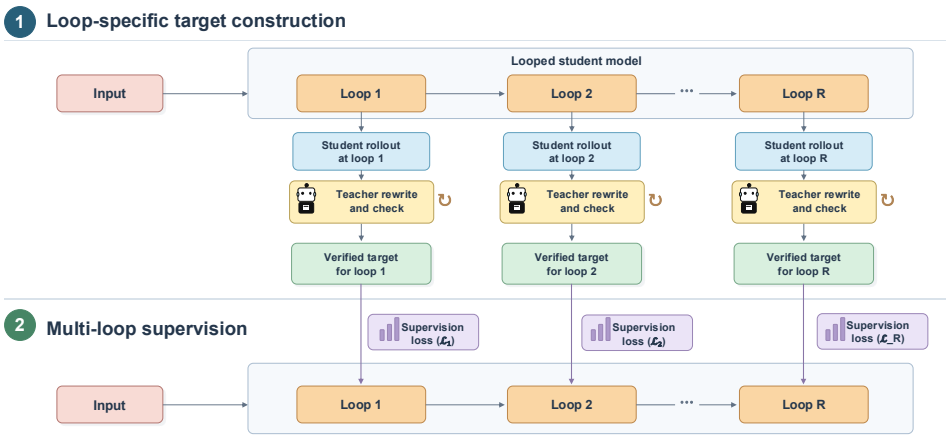


Figure 2: Overview of LoopDistill. The student generates trajectories at fixed recurrent depths. The rewrite harness corrects and verifies them using teacher guidance. Each accepted target supervises the student's readout at the corresponding depth.

3 Proposed Method

3.1 Preliminary

A Looped Language Model (LoopLM) repeatedly applies the same Transformer layers before predicting a token (Zhu et al., 2025). Let $\mathcal{V}$ be the vocabulary, d the hidden dimension, and $z_{1:M}$ a sequence of M tokens. Denote the token embedding by $\mathrm{emb} : \mathcal{V} \to \mathbb{R}^d$ and a stack of L causal Transformer layers by

$$\mathcal{M}_\theta = \mathcal{T}_{\theta_L} \circ \cdots \circ \mathcal{T}_{\theta_1},$$

where θ=(θ_1, ..., θ_L) denotes the parameters of the L-layer Transformer stack. Starting from the embedded sequence $H^{(0)} = \mathrm{emb}(z_{1:M})$, the model computes

$$H^{(r)} = \mathcal{M}_\theta(H^{(r-1)}), \qquad r = 1, \ldots, R, \tag{1}$$

where R is the maximum recurrent depth. The same parameters θ are reused at every loop. Thus, $r = 1$ corresponds to one pass through the L-layer stack, whereas $r > 1$ performs additional latent computation with that stack.

Each depth has a next-token prediction through the shared language modeling head. If $h_\ell^{(r)}$ is the hidden state at position ℓ after r loops, then

$$p_\theta^{(r)}(\cdot \mid z_{1:\ell}) = \mathrm{softmax}\Big(\mathrm{lmhead}(h_\ell^{(r)})\Big), \tag{2}$$

where $\mathrm{lmhead} : \mathbb{R}^d \to \mathbb{R}^{|\mathcal{V}|}$. A next-token loss can therefore be computed at each recurrent depth:

$$\mathcal{L}_{\mathrm{LM}}^{(r)} = -\sum_{\ell=1}^{M-1} \log p_\theta^{(r)}(z_{\ell+1} \mid z_{1:\ell}).$$

3.2 Overview

LoopDistill follows three stages, illustrated in Figure 2 and summarized in Algorithm 1. For each training problem x and recurrent depth $r \in \{1, \ldots, R\}$, we first fix the student's readout to depth r at every decoding step and generate a complete trajectory $\hat{y}^{(r)}$, which we call the *raw student rollout*. Separately, the teacher produces an original solution trajectory y^T. The rewrite harness uses x, y^T, and $\hat{y}^{(r)}$ to construct a corrected, student-compatible trajectory $\tilde{y}^{(r)}$ It accepts the rewrite only after checking solution quality and compatibility with the raw rollout Finally, each accepted $\tilde{y}^{(r)}$ supervises the student readout at depth r. Thus, the student rollout supplies the depth-specific reference, while the verified rewrite supplies the training target.

3.3 PER-LOOP STUDENT-AWARE TRAJECTORY REWRITING

Depth-specific student rollouts. For each problem x, we generate a raw student rollout $\hat{y}^{(r)}$ at recurrent depth r. We fix the exit depth to r for every generated token, obtaining

$$\hat{y}^{(r)} \sim P_\theta^{(r)}(\cdot \mid x), \qquad r = 1, \ldots, R, \tag{3}$$

where $P_\theta^{(r)}$ is the autoregressive distribution over complete trajectories induced by the depth-r next-token distribution $p_\theta^{(r)}$ defined in Equation (2). The collection $\{\hat{y}^{(r)}\}_{r=1}^{R}$ exposes how the student's generated reasoning varies with recurrent depth, including differences in length, organization, exploration, and self-correction.

Let y^T denote the teacher's original reasoning trajectory for x, whose final solution passes the task verifier. For each depth r, the teacher uses y^T as a solution reference and $\hat{y}^{(r)}$ as a behavioral reference to propose a corrected We denote a rewrite that passes the verification harness by $\tilde{y}^{(r)}$. Unlike $\hat{y}^{(r)}$, this accepted rewrite is used as the supervision target for loop r.

Reasoning profiles. We analyze each raw student rollout $\hat{y}^{(r)}$ using LCoT2Tree (Jiang et al., 2025). LCoT2Tree extracts a reasoning sketch, splits the trajectory into contiguous thought units, aligns the units with the sketch, and labels their relationships to preceding reasoning. It then organizes the units into a reasoning tree. For depth r, let

$$\big(\mathcal{S}^{(r)}, \mathcal{G}^{(r)}\big) = \mathrm{LCoT2Tree}\big(\hat{y}^{(r)}\big), \qquad \mathcal{S}^{(r)} = \big\{(s_i^{(r)}, a_i^{(r)})\big\}_{i=1}^{m_r}, \tag{4}$$

where $s_i^{(r)}$ is a thought unit, $\mathcal{G}^{(r)}$ is the reasoning tree, and $a_i^{(r)}$ labels the unit as continuous logic, exploration, backtracking, or verification. The label describes the unit's role in the reasoning process, not its factual correctness.

We summarize the rollout by $\psi^{(r)} = (\ell^{(r)}, b^{(r)}, \boldsymbol{\rho}^{(r)}, M^{(r)})$. Here, $\ell^{(r)} = |\hat{y}^{(r)}|$ is the token length of the raw rollout; $b^{(r)}$ is the number of exploratory branches in $\mathcal{G}^{(r)}$; $\boldsymbol{\rho}^{(r)}$ gives the fraction of reasoning tokens assigned to each role; and $M^{(r)}$ contains salient discourse markers. This profile describes both how much reasoning the student generates at depth r and how that reasoning unfolds.

Trajectory Adaptation. A single editing instruction is unsuitable for every student trajectory. We therefore choose among three operations based on the student's profiles:

1. **Whole-trajectory correction.** For a short or otherwise coherent student rollout, the teacher receives x, y^T, and the full $\hat{y}^{(r)}$. It corrects the trajectory as a whole, retaining its useful reasoning and characteristic phrasing.
2. **Segment-wise correction.** For a long but interpretable rollout with substantial exploration or backtracking, the teacher receives the full trajectory together with $\mathcal{S}^{(r)}$. It reviews the units in order, corrects mistaken steps in place, and consolidates redundant passages while preserving the sequence of meaningful attempts.
3. **Prefix continuation.** For a long rollout with a repetitive or otherwise unproductive suffix, we select a coherent, nondegenerate prefix $\hat{y}_{1:k_r}^{(r)}$. The teacher receives this prefix and continues from it instead of editing the unproductive suffix. If the prefix contains a meaningful false start, the continuation must explicitly correct it.

The operation is chosen for each teacher–student pair and recurrent depth. In particular, segment-wise correction retains the structure of a recoverable exploratory trace, whereas prefix continuation avoids using a degenerate tail as a template. Denoting the selected operation by o_r and the rewrite operation by $\mathcal{R}_T$, the teacher proposes a candidate

$$\tilde{y}^{(r)} = \mathcal{R}_T^{(o_r)}\big(x, y^T, \hat{y}^{(r)}, \mathcal{S}^{(r)}, \psi_r\big), \tag{5}$$

where each operation exposes only its specified student content to the teacher. Thus, the prefix-continuation operation uses the selected prefix rather than the full rollout. The example prompts for the rewriting strategy are provided in Section A.

Algorithm 1 LoopDistill: Per-Loop Rewriting and Multi-Loop Supervision

Require: Training problems $\mathcal{D}$, initial student parameters θ_0, teacher T, supervised depths $\mathcal{R}_{\text{sup}} \subseteq \{1, \ldots, R\}$, maximum rewrite attempts $K_{\max}$, fixed loss coefficients $\{\lambda_r\}_{r \in \mathcal{R}_{\text{sup}}}$
Ensure: Distilled student parameters θ

$\mathcal{C} \leftarrow \varnothing$ ▷ Construct verified training examples
for $x \in \mathcal{D}$ **do**
  $y^T \leftarrow \textsc{GenerateTeacher}(T, x)$
  if $\neg \textsc{VerifySolution}(x, y^T)$ **then**
    continue
  end if
  $\mathcal{Y}(x) \leftarrow \varnothing$
  for $r \in \mathcal{R}_{\text{sup}}$ **do**
    $\hat{y}^{(r)} \leftarrow \textsc{GenerateAtDepth}(p_{\theta_0}^{(r)}, x)$
    $(\mathcal{S}^{(r)}, \mathcal{G}^{(r)}) \leftarrow \textsc{LCoT2Tree}(\hat{y}^{(r)})$
    $\psi^{(r)} \leftarrow \textsc{ReasoningProfile}(\hat{y}^{(r)}, \mathcal{S}^{(r)}, \mathcal{G}^{(r)})$
    $o_r \leftarrow \textsc{SelectRewriteStrategy}(\psi^{(r)}, \hat{y}^{(r)})$ ▷ Whole, segment-wise, or prefix
    $f \leftarrow \varnothing$
    for $k = 1, \ldots, K_{\max}$ **do**
      $\tilde{y}_k^{(r)} \leftarrow \textsc{TeacherRewrite}(T, x, y^T, \hat{y}^{(r)}, \mathcal{S}^{(r)}, o_r, f)$
      $(a, f) \leftarrow \textsc{VerifyRewrite}(x, \tilde{y}_k^{(r)}, \hat{y}^{(r)}, \psi^{(r)})$ ▷ Quality and student compatibility
      if $a = \text{true}$ **then**
        $\mathcal{Y}(x) \leftarrow \mathcal{Y}(x) \cup \{(r, \tilde{y}_k^{(r)})\}$
        break
      end if
    end for
  end for
  if $\mathcal{Y}(x) \neq \varnothing$ **then**
    $\mathcal{C} \leftarrow \mathcal{C} \cup \{(x, \mathcal{Y}(x))\}$
  end if
end for
$\theta \leftarrow \theta_0$ ▷ Train on accepted depth–rewrite pairs
while training has not converged **do**
  $\mathcal{B} \leftarrow \textsc{SampleBatch}(\mathcal{C})$
  $\mathcal{L} \leftarrow \dfrac{1}{|\mathcal{B}|} \displaystyle\sum_{(x, \mathcal{Y}(x)) \in \mathcal{B}} \mathcal{L}_{\text{LD}}\big(x, \mathcal{Y}(x); \theta\big)$ ▷ Defined in Equation (8)
  Update θ using $\nabla_\theta \mathcal{L}$
end while
return θ

3.4 VERIFICATION

A correct final answer alone does not make an adaptation suitable for distillation. The verification therefore checks both solution quality and compatibility with the student rollout.

Solution quality. A task-specific verifier checks the final answer or executable output. We also require the rewrite to be complete and clean: it must reach a finished solution without unresolved errors or degenerate repetition. The teacher may retain an incorrect student attempt, but must identify and correct it before the final answer.

Student compatibility. An unconstrained teacher rewrite can collapse an exploratory student trajectory into a short, textbook-style solution. We therefore compare each candidate rewrite $\tilde{y}^{(r)}$ with the raw student rollout $\hat{y}^{(r)}$ at the same recurrent depth. Recall that $\ell^{(r)} = |\hat{y}^{(r)}|$ and $b^{(r)}$ are the rollout's token length and number of exploratory branches. Let $\tilde{\ell}^{(r)} = |\tilde{y}^{(r)}|$, and let $\tilde{b}^{(r)}$ be the number of exploratory branches in the rewrite, measured using the same LCoT2Tree procedure. We require

$$\tau_{\text{len}}^{-} \leq \frac{\tilde{\ell}^{(r)}}{\ell^{(r)}} \leq \tau_{\text{len}}^{+}, \qquad \tau_{\text{branch}}^{-} \leq \frac{\tilde{b}^{(r)}}{b^{(r)}} \leq \tau_{\text{branch}}^{+}. \tag{6}$$

Here, τ^{-}_{len} and τ^{+}_{len} are the lower and upper bounds on the rewrite-to-student length ratio, while τ^{-}_{branch} and τ^{+}_{branch} are the corresponding bounds on the exploratory-branch ratio. The branch-ratio condition applies only when $b^{(r)} > 0$. The length bounds prevent excessive compression or expansion, while the branch bounds preserve the extent of the student's exploration. We additionally check that branches in the rewrite correspond to recognizable student attempts rather than unrelated detours. An incorrect attempt may be retained if the rewrite identifies and resolves its error. The teacher is also instructed to retain characteristic discourse markers from $M^{(r)}$ where they fit naturally.

If a candidate fails a check, the harness returns the rejection reason to the teacher for revision. We allow up to three proposals per depth and discard candidates that still fail.

3.5 Multi-Loop Supervision

A looped language model provides a next-token distribution $p_\theta^{(r)}$ after each recurrent depth r. LoopLM uses these readouts for per-loop language-model training (Zhu et al., 2025). In LoopDistill, we select supervised depths $\mathcal{R}_{\text{sup}} \subseteq \{1, \dots, R\}$ and construct a distinct reasoning target from the student rollout at each selected depth.

For a problem x, let $\mathcal{Y}(x)$ be the set of accepted depth–rewrite pairs $(r, \tilde{y}^{(r)})$. For each pair, we teacher-force the rewrite and evaluate its next-token loss using the readout after exactly r loops:

$$\mathcal{L}_{\text{KD}}^{(r)}\big(x, \tilde{y}^{(r)}; \theta\big) = -\frac{1}{\tilde{\ell}^{(r)}} \sum_{t=1}^{\tilde{\ell}^{(r)}} \log p_\theta^{(r)}\Big(\tilde{y}_t^{(r)} \mid x, \tilde{y}_{<t}^{(r)}\Big), \tag{7}$$

where $\tilde{\ell}^{(r)} = |\tilde{y}^{(r)}|$ and $p_\theta^{(r)}$ is the depth-r distribution defined in Equation (2). Both the target tokens and their teacher-forced prefixes come from the rewrite assigned to depth r. Normalizing by $\tilde{\ell}^{(r)}$ prevents longer rewrites from receiving greater weight solely because they contain more tokens.

We combine the losses over accepted rewrites:

$$\mathcal{L}_{\text{LD}}\big(x, \mathcal{Y}(x); \theta\big) = \frac{\sum_{(r,\tilde{y}^{(r)}) \in \mathcal{Y}(x)} \lambda_r \, \mathcal{L}_{\text{KD}}^{(r)}\big(x, \tilde{y}^{(r)}; \theta\big)}{\sum_{(r,\tilde{y}^{(r)}) \in \mathcal{Y}(x)} \lambda_r}, \tag{8}$$

where $\lambda_r > 0$ is a fixed loss coefficient for depth r; only θ contains trainable model parameters. We exclude a problem when $\mathcal{Y}(x) = \varnothing$, but an accepted rewrite can still provide supervision if rewriting fails at another selected depth.

Each accepted $\tilde{y}^{(r)}$ corrects the corresponding raw student rollout $\hat{y}^{(r)}$ while retaining its recognizable reasoning pattern. The targets need not become progressively longer or more elaborate with depth: each supervised readout learns from a trajectory adapted to the reasoning it generates.

4 Experiments

We evaluate LoopDistill on mathematical reasoning datasets, using Ouro-2.6B-Base as the looped student. We distill from two non-looped teachers, Qwen3-30B-A3B and gpt-oss-20b, to examine the method under different teacher reasoning patterns. Our experiments address two questions: (1) Does LoopDistill improve reasoning and code generation in a looped student? (2) Does student-aware, loop-specific rewriting improve upon direct distillation of teacher-generated trajectories?

4.1 Experimental Setup

Student and teacher models. We use Ouro-2.6B-Base (Zhu et al., 2025) as the student in all experiments. For each training problem, we generate student trajectories at fixed recurrent depths and construct teacher rewrites using the verification harness in Sections 3.3 and 3.4. Accepted rewrites supervise their corresponding loops as described in Section 3.5. We repeat this procedure separately with Qwen3-30B-A3B (Yang et al., 2025) and gpt-oss-20b (OpenAI et al., 2025) as teachers.

Table 1: Main results with Ouro-2.6B-Base as the student. Math models are trained on DAPO-Math-8k-Stratified. We report math accuracy in %.

		Mathematical reasoning			
Teacher	**Method**	**AIME24**	**AIME25**	**AIME26**	**OlympiadBench**
–	Ouro-2.6B-Base	33.3	30.0	30.0	53.1
Qwen3-30B-A3B	KD	36.7	34.4	38.9	61.1
	OPD	48.9	41.1	32.2	56.4
	TRD	46.7	36.7	40.0	59.7
	LoopDistill (Ours)	**56.7**	**47.8**	**58.9**	**66.5**
gpt-oss-20b	KD	20.0	23.3	18.9	48.9
	OPD	12.2	20.0	16.7	37.3
	TRD	32.2	**35.6**	35.6	**57.6**
	LoopDistill (Ours)	**40.0**	31.1	**46.7**	57.2

Mathematical reasoning datasets. We use DAPO-Math-8k-Stratified, an 8,000-problem subset of DAPO-Math, for training (Yu et al., 2025). We train on its 7,500-example training split and reserve its 500-example validation split. We evaluate on AIME24, AIME25, AIME26, and the English, text-only competition mathematics subset of OlympiadBench (`OE_TO_maths_en_COMP`) (He et al., 2024). We report answer accuracy using the answer extraction and grading procedure for each benchmark.

Baselines. We compare LoopDistill with three distillation baselines. **KD** trains the student by teacher-forced cross-entropy on complete teacher-generated reasoning trajectories (Kim & Rush, 2016). **OPD** samples student trajectories and uses teacher feedback at the sampled prefixes (Agarwal et al., 2024a). For our OPD baseline, we use Hugging Face's GOLD implementation to handle the teacher's and student's different tokenizers. **TRD** asks the teacher to refine a student rollout to repair problematic prefixes, then applies token-level distillation along the refined trajectory (Jiang et al., 2026). TRD provides a particularly relevant comparison for trajectory rewriting; our method additionally constructs and verifies targets for selected recurrent depths and supervises their corresponding readouts.

Training and evaluation setup. The student model is trained with full fine-tuning for 3 epochs at a peak learning rate of 2×10^{-5} under a cosine schedule with a warmup ratio of 0.1. The supervision is on loop 3 and loop 4 of the student. We use a micro-batch of 1 with 128 gradient-accumulation steps, giving a global batch of 128 sequences, and cap sequences at $16{,}384$ tokens with dynamic padding to the per-batch maximum. At evaluation time we decode with $16{,}384$ new tokens, a prompt budget of $1{,}024$ tokens, temperature 0.6, top-p 0.95, top-k 0 and the readout loop is at loop 4. We draw $n = 3$ samples per problem and report avg@3. All training and evaluation runs on a single node with 8 NVIDIA H200 GPUs.

4.2 Main Results

Teacher trajectories can produce negative transfer. Table 1 shows that direct KD depends strongly on the teacher. With Qwen3-30B-A3B, KD improves over the unadapted Ouro-2.6B-Base on all four benchmarks. With gpt-oss-20b, however, KD falls below the unadapted student on every benchmark: AIME24 drops from 33.3% to 20.0%, and AIME26 from 30.0% to 18.9%. Thus, trajectories from a substantially larger teacher can be harmful as direct training targets. This result motivates evaluating whether supervision adapted to student rollouts improves knowledge transfer.

Student-aware supervision restores positive transfer. With gpt-oss-20b, LoopDistill exceeds the unadapted student on all four benchmarks, reversing the negative transfer observed with direct KD. With Qwen3-30B-A3B, it achieves the highest accuracy on all four benchmarks. The largest gains for both teachers occur on AIME26, suggesting that this benchmark benefits particularly from aligning targets with the student's recurrent computation. The benefit is not uniform: TRD remains stronger than LoopDistill with gpt-oss-20b on AIME25 and marginally stronger on OlympiadBench.

Table 2: Ablation study with Ouro-2.6B-Base as the student, comparing Qwen3-30B-A3B and GPT-OSS-20B as teachers. Math uses DAPO-Math-8k-Stratified. We report math accuracy in %.

		Mathematical reasoning			
Trajectory	**Supervision**	**AIME24**	**AIME25**	**AIME26**	**OlympiadBench**
Teacher: Qwen3-30B-A3B					
Teacher	Last-loop	36.7	34.4	38.9	61.1
Teacher	Two-loop	44.4	34.4	41.1	59.5
Rewritten Student	Last-loop	40.0	41.1	45.6	66.3
Rewritten Student	Two-loop	**56.7**	**47.8**	**58.9**	**66.5**
Teacher: GPT-OSS-20B					
Teacher	Last-loop	20.0	23.3	18.9	48.9
Teacher	Two-loop	28.9	23.3	27.8	51.1
Rewritten Student	Last-loop	**42.2**	30.0	33.3	**57.2**
Rewritten Student	Two-loop	40.0	**31.1**	**46.7**	**57.2**

Together, the results indicate that the suitability of the distillation trajectory matters alongside the choice of teacher.

4.3 Ablation Study

Rewriting improves the training target. Holding last-loop supervision fixed, replacing the original teacher trajectory with a verified, student-aware rewrite improves accuracy on every benchmark with both teachers (Table 2). The effect is particularly large for gpt-oss-20b: AIME24 accuracy rises from 20.0% to 42.2%, and AIME26 accuracy rises from 18.9% to 33.3%. Because the supervised loop is unchanged, these gains isolate the contribution of target construction from the contribution of supervising an additional loop. They show that a corrected, student-derived trajectory can be a more effective target than this teacher's original trajectory.

Intermediate-loop supervision depends on the target. Adding two-loop supervision to the original teacher trajectories yields mixed results. It improves AIME24 and AIME26 with both teachers, but does not improve AIME25 and reduces OlympiadBench accuracy with Qwen3-30B-A3B. When the targets are loop-specific rewrites, two-loop supervision provides further gains on AIME26 with both teachers and on AIME24 and AIME25 with Qwen3-30B-A3B, although AIME24 accuracy decreases slightly with gpt-oss-20b. Thus, supervising an intermediate loop offers a distinct benefit, but adding supervision alone does not reliably produce the gains observed with rewritten targets.

5 Conclusion and Limitations

We introduced LoopDistill, a student-aware distillation framework for looped language models. It uses student rollouts at selected recurrent depths to guide teacher rewrites, verifies the resulting trajectories, and supervises each selected loop with its corresponding target. On four mathematical reasoning benchmarks, LoopDistill improves over direct distillation with both teachers, suggesting that reasoning supervision benefits from accounting for the student's behavior across recurrent depth.

While our verification checks the final answer and trajectory compatibility, it cannot establish that every intermediate reasoning step is valid. Finally, we construct targets for individual loops without explicitly constraining how reasoning should evolve between them. Constructing separate targets also increases teacher-generation cost.

AI Use Statement

We used generative AI models to generate and rewrite reasoning trajectories used as training targets. We also used an AI assistant for manuscript editing and figure development. The authors reviewed the AI-assisted work and take responsibility for the final content, claims, and artifacts in this paper.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes, 2024a. URL https://arxiv.org/abs/2306.13649.

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

Sangmin Bae, Adam Fisch, Hrayr Harutyunyan, Ziwei Ji, Seungyeon Kim, and Tal Schuster. Relaxed recursive transformers: Effective parameter sharing with layer-wise lora. In *International Conference on Learning Representations*, volume 2025, pp. 34282–34327, 2025.

Ying Fan, Anej Svete, and Kangwook Lee. Bridging the gap between latent and explicit reasoning with looped transformers. *arXiv preprint arXiv:2606.31779*, 2026.

Zitian Gao, Yilong Chen, Yihao Xiao, Xinyu Yang, Ran Tao, Joey Zhou, and Bryan Dai. Loop the loopies!, 2026. URL https://arxiv.org/abs/2607.16051.

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

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. OlympiadBench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3828–3850, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.211. URL https://aclanthology.org/2024.acl-long.211/.

Gangwei Jiang, Yahui Liu, Zhaoyi Li, Wei Bi, Fuzheng Zhang, Linqi Song, Ying Wei, and Defu Lian. What makes a good reasoning chain? uncovering structural patterns in long chain-of-thought reasoning. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 6490–6514, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.329. URL https://aclanthology.org/2025.emnlp-main.329/.

Li Jiang, Haoran Xu, Yichuan Ding, and Amy Zhang. Trajectory-refined distillation, 2026. URL https://arxiv.org/abs/2606.08432.

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.

Jongwoo Ko, Sungnyun Kim, Tianyi Chen, and Se-Young Yun. Distillm: Towards streamlined distillation for large language models. *arXiv preprint arXiv:2402.03898*, 2024.

Yuetai Li, Xiang Yue, Zhangchen Xu, Fengqing Jiang, Luyao Niu, Bill Yuchen Lin, Bhaskar Ramasubramanian, and Radha Poovendran. Small models struggle to learn from strong reasoners, 2025. URL https://arxiv.org/abs/2502.12143.

Lucie Charlotte Magister, Jonathan Mallinson, Jakub Adamek, Eric Malmi, and Aliaksei Severyn. Teaching small language models to reason. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 1773–1781, 2023.

OpenAI, :, Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K. Arora, Yu Bai, Bowen Baker, Haiming Bao, Boaz Barak, Ally Bennett, Tyler Bertao, Nivedita Brett, Eugene Brevdo, Greg Brockman, Sebastien Bubeck, Che Chang, Kai Chen, Mark Chen, Enoch Cheung, Aidan Clark, Dan Cook, Marat Dukhan, Casey Dvorak, Kevin

Fives, Vlad Fomenko, Timur Garipov, Kristian Georgiev, Mia Glaese, Tarun Gogineni, Adam Goucher, Lukas Gross, Katia Gil Guzman, John Hallman, Jackie Hehir, Johannes Heidecke, Alec Helyar, Haitang Hu, Romain Huet, Jacob Huh, Saachi Jain, Zach Johnson, Chris Koch, Irina Kofman, Dominik Kundel, Jason Kwon, Volodymyr Kyrylov, Elaine Ya Le, Guillaume Leclerc, James Park Lennon, Scott Lessans, Mario Lezcano-Casado, Yuanzhi Li, Zhuohan Li, Ji Lin, Jordan Liss, Lily, Liu, Jiancheng Liu, Kevin Lu, Chris Lu, Zoran Martinovic, Lindsay McCallum, Josh McGrath, Scott McKinney, Aidan McLaughlin, Song Mei, Steve Mostovoy, Tong Mu, Gideon Myles, Alexander Neitz, Alex Nichol, Jakub Pachocki, Alex Paino, Dana Palmie, Ashley Pantuliano, Giambattista Parascandolo, Jongsoo Park, Leher Pathak, Carolina Paz, Ludovic Peran, Dmitry Pimenov, Michelle Pokrass, Elizabeth Proehl, Huida Qiu, Gaby Raila, Filippo Raso, Hongyu Ren, Kimmy Richardson, David Robinson, Bob Rotsted, Hadi Salman, Suvansh Sanjeev, Max Schwarzer, D. Sculley, Harshit Sikchi, Kendal Simon, Karan Singhal, Yang Song, Dane Stuckey, Zhiqing Sun, Philippe Tillet, Sam Toizer, Foivos Tsimpourlas, Nikhil Vyas, Eric Wallace, Xin Wang, Miles Wang, Olivia Watkins, Kevin Weil, Amy Wendling, Kevin Whinnery, Cedric Whitney, Hannah Wong, Lin Yang, Yu Yang, Michihiro Yasunaga, Kristen Ying, Wojciech Zaremba, Wenting Zhan, Cyril Zhang, Brian Zhang, Eddie Zhang, and Shengjia Zhao. gpt-oss-120b & gpt-oss-20b model card, 2025. URL https://arxiv.org/abs/2508.10925.

Andrei Cristian Popescu, Haitz Sáez de Ocáriz Borde, and Pietro Liò. Looped language models improve compositional tool calling, 2026. URL https://arxiv.org/abs/2608.18171.

Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J Reddi. Reasoning with latent thoughts: On the power of looped transformers. In *International Conference on Learning Representations*, volume 2025, pp. 14855–14881, 2025.

Kumar Shridhar, Alessandro Stolfo, and Mrinmaya Sachan. Distilling reasoning capabilities into smaller language models. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 7059–7073, 2023.

Jonathan Williams and Esin Tureci. Prioritize the process, not just the outcome: Rewarding latent thought trajectories improves reasoning in looped language models. *arXiv preprint arXiv:2602.10520*, 2026.

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, 2025. URL https://arxiv.org/abs/2505.09388.

Jian Yang, Shawn Guo, Wei Zhang, Tianyu Zheng, Yaxin Du, Haau-Sing Li, Jiajun Wu, Yue Song, Yan Xing, Qingsong Cai, Zelong Huang, Chuan Hao, Ran Tao, Xianglong Liu, Wayne Xin Zhao, Mingjie Tang, Weifeng Lv, Ming Zhou, and Bryan Dai. Loopcoder-v2: Only loop once for efficient test-time computation scaling, 2026a. URL https://arxiv.org/abs/2606.18023.

Yuming Yang, Mingyoung Lai, Wanxu Zhao, Xiaoran Fan, Zhiheng Xi, Mingqi Wu, Chiyue Huang, Jun Zhao, Haijun Lv, Jian Tong, Yunhua Zhou, Yicheng Zou, Qipeng Guo, Tao Gui, Qi Zhang, and Xuanjing Huang. Which reasoning trajectories teach students to reason better? a simple metric of informative alignment, 2026b. URL https://arxiv.org/abs/2601.14249.

Qifan Yu, Zhenyu He, Sijie Li, Zhou Xun, Jun Zhang, Jingjing Xu, and Di He. Enhancing autoregressive chain-of-thought through loop-aligned reasoning. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2206–2222, 2026.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu,

Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. Dapo: An open-source llm reinforcement learning system at scale, 2025. URL https://arxiv.org/abs/2503.14476.

Rui-Jie Zhu, Zixuan Wang, Kai Hua, Tianyu Zhang, Ziniu Li, Haoran Que, Boyi Wei, Zixin Wen, Fan Yin, He Xing, et al. Scaling latent reasoning via looped language models. *arXiv preprint arXiv:2510.25741*, 2025.

A PROMPTS

PREFIX-CONTINUATION PROMPTS

System message

```
You continue half-written working.  The text you are given
ends mid-thought and you carry it on in the same voice it is
already in:  informal, first person, working things out on
the page.

You never restructure it.  No section headings, no numbered
steps, no restating the problem, no converting inline
arithmetic into display equations, no complexity analysis.
Messy is correct here --- doubling back, noticing an error
and fixing it in place, saying ``wait, that's wrong'' are
all part of the voice.

You show work rather than describe it.  Writing out the
algebra is the task; commenting on whether the work so far
is good, or referring to whoever wrote it, is not.
```

User message for math.

```
Solve this problem.  Work in first person, think step by
step, and show your working as you go --- including any
wrong turns you take and how you recover from them.

End with the final answer in \boxed{} and STOP IMMEDIATELY
after it.  Nothing may follow the box.

Problem:
{problem}
```

User message for code.

```
Solve this problem.  Work in first person, think step by
step, and show your working as you go --- including any
wrong turns you take and how you recover from them.

Finish with the complete solution as a single,
self-contained Python module in a fenced code block
(```python ... ```), including every import and helper it
needs to run on its own.  Keep the reasoning as prose before
it.  The module must define `{entry}` with exactly that
name, because the tests import it by name.

Problem:
{problem}
```

Length note.

```
You have already written about {n_done} characters of this.
Carrying it through to the answer at the same level of
detail should take roughly {n_min}--{n_max} more characters
of actual working.  Much less than that means you summarised
instead of continuing.
```