Looped Language Model Latent Reasoning Compression via On-Policy Self-Distillation

Jon Williams

February 13, 2026

Abstract

Looped language models (LoopLMs) perform multiple latent "thinking" iterations per token, producing intermediate next-token distributions before a final next-token distribution. On-policy self-distillation methods such as OPSD and SDPO replace sparse supervision with dense token-level distribution matching between a student and a privileged-context teacher. This proposal aims to leverage on-policy self-distillation to compress the latent reasoning of LoopLMs. We specifically match student next token distributions at earlier loops to teacher next token distributions at later loops. We hypothesize that distilling the more refined latent thinking of the privileged-context teacher to the less-refined latent thinking of the student can improve the latent reasoning efficiency of the student.

1 Motivation

- Distillation has shown to outperform RLVR due to giving richer feedback
- LoopLMs have been shown to outperform traditional LLMs
- RLTT demonstrated that post-training techniques which operate on the entire loop-level lead to better results than those that operate on just the terminal loop level.
- It would be nice if a LoopLM could loop for fewer iterations but match the performance gained by looping for more iterations.
  - Suppose we can get similar performance attained by looping for 4 iterations by looping for only 3 iterations. We would have effectively cut inference cost by 25% *for free*.

Key question: Can we compress LoopLM latent reasoning by *distilling a teacher's later latent thiking into a students earlier latent thinking*?

2 Related Work

2.1 On-Policy Self-Distillation

OPSD trains a student policy to match a teacher distribution produced by the same model conditioned on privileged information (e.g., ground-truth answer or verified trace), computed on student on-policy rollouts [4]. **SDPO** similarly constructs a teacher by re-conditioning the model on feedback (e.g., unit test failures, critique) to produce dense token-level signals in place of sparse rewards [3].

2.2 Looped Language Models

LoopLMs (Ouro-style) execute latent recurrent processing steps before emitting each token, producing intermediate distributions $p^{(1)}, \ldots, p^{(R)}$ before sampling from the final step [1]. This creates an explicit latent reasoning trajectory.

2.3 RLTT

RLTT suggests that distributing learning signal across loop iterations (rather than only the final loop) can substantially improved reasoning [2].

3 Proposed Method: Latent-Loop On-Policy Self-Distillation (LLOPSD)

3.1 Setup and Notation

Let $\mathcal{D} = \{(x, a)\}$ be a dataset of prompts x with ground-truth answers a. A LoopLM can run up to R latent loop iterations per token. In this work, we explicitly train a *compressed student* that runs only $\tilde{R}$ loops per token at inference, where $\tilde{R} \leq R$, while the teacher runs the full R loops.

For a prefix $y_{<t}$ and prompt x:

- **Student per-loop distribution (compressed compute):**

$$p^{(r)}_{\theta,S}(\cdot \mid x, y_{<t}) \in \mathbb{R}^{|V|}, \quad r \in \{1, \ldots, \tilde{R}\}.$$

- **Teacher per-loop distribution (privileged conditioning, full compute):**

$$p^{(r)}_{\theta',T}(\cdot \mid x, c, y_{<t}), \quad r \in \{1, \ldots, R\},$$

  where θ' may be either the exact student params θ an EMA copy of the student params θ, or a frozen teacher, and c is privileged teacher context.

We generate an on-policy rollout using the *student's* final loop:

$$y \sim p^{(\tilde{R})}_{\theta,S}(\cdot \mid x).$$

3.2 Loop Compression Distillation Objective

Standard on-policy distillation matches teacher/student distributions token-wise under the same computation depth. Here, our goal is **latent reasoning compression**: we match *earlier* student loop distributions to *later* teacher loop distributions so that the student can run fewer loops while approaching the teacher's R-loop performance.

Define a **loop mapping** function

$$\phi : \{1, \ldots, \tilde{R}\} \to \{1, \ldots, R\},$$

with the constraints that ϕ is non-decreasing and $\phi(r) \geq r$ (student loop r distills toward a teacher loop that is at least as refined). A simple and effective default is the **shift mapping**:

$$\phi(r) = r + (R - \tilde{R}),$$

e.g., if $(R, \tilde{R}) = (4, 2)$ then $\phi(1) = 3$ and $\phi(2) = 4$.

Our cross-loop on-policy distillation loss is:

$$L_{\text{loop-distill}}(x, y) = \frac{1}{|y|} \sum_{t=1}^{|y|} \sum_{r=1}^{\tilde{R}} \alpha_r \cdot D\Big(p_{\theta,S}^{(r)}(\cdot \mid x, y_{<t}) \,\Big\|\, \texttt{stopgrad}(p_{\theta',T}^{(\phi(r))}(\cdot \mid x, c, y_{<t}))\Big), \tag{1}$$

where:

- $D(\cdot\|\cdot)$ is a divergence (e.g., forward KL, reverse KL, or JSD).
- $\alpha_r \geq 0$ and $\sum_{r=1}^{\tilde{R}} \alpha_r = 1$ (convex combination across *student* loops).
- $\texttt{stopgrad}(\cdot)$ blocks gradients through the teacher distribution.

Design choices. We expect both ϕ and $\{\alpha_r\}$ to matter for stability and compression quality. We will evaluate:

1. **Terminal-only compression baseline:** $\alpha_{\tilde{R}} = 1$ and $\phi(\tilde{R}) = R$ (match only student final loop to teacher final loop).
2. **Uniform:** $\alpha_r = 1/\tilde{R}$.
3. **Late-heavy over student loops:** $\alpha_r \propto r^\gamma$ for $\gamma > 0$ (prioritize aligning the student's later loops to the teacher's later loops).
4. **Mapping ablations:** shift mapping (e.g $1 \to 3$, $2 \to 4$) vs. linear mapping (e.g $1 \to 2$, $2 \to 4$) vs. fixed targets (e.g $1 \to 4$, $2 \to 4$).

3.3 Teacher Privileged Contexts

Our method can be instantiated with different ways of creating c:

- **OPSD-style:** $c \leftarrow a$ (ground-truth answer or verified trace).
- **SDPO-style:** $c \leftarrow f$ (feedback/judge critique/tests from the rollout).

This yields a teacher distribution that is strictly more informed than the student at training time.

4 Algorithm

Algorithm 1 Latent-Loop On-Policy Self-Distillation (LLOPSD)

Require: Dataset $\mathcal{D}$, student LoopLM π_θ, teacher LoopLM $\pi_{\theta'}$
Require: Teacher loops R, student loops $\tilde{R} \le R$, divergence $D(\cdot\|\cdot)$
Require: Weights $\alpha_{1..\tilde{R}}$, loop mapping $\phi(\cdot)$
Require: Teacher context constructor $\texttt{ctx}(x, y)$ (e.g. a or f as in OPSD/SDPO)

for each minibatch $B \subset \mathcal{D}$ **do**
  $L_{\text{batch}} \leftarrow 0$
  for each sample x in B **do**
    $L_{\text{sample}} \leftarrow 0$
    Sample rollout $y \sim \pi_\theta(\cdot \mid x)$ using the **student's final-loop** distribution $\tilde{R}$
    $c \leftarrow \texttt{ctx}(x, y)$ ▷ privileged teacher conditioning
    for $t = 1$ to $|y|$ **do**
      for $r = 1$ to $\tilde{R}$ **do**
        $r_T \leftarrow \phi(r)$
        $p_S^{(r)}(\cdot) \leftarrow p_{\theta,S}^{(r)}(\cdot \mid x, y_{<t})$
        $p_T^{(r_T)}(\cdot) \leftarrow p_{\theta',T}^{(r_T)}(\cdot \mid x, c, y_{<t})$
        $L_{\text{sample}} \leftarrow L_{\text{sample}} + \alpha_r \cdot D\Big(p_S^{(r)} \,\|\, \texttt{stopgrad}(p_T^{(r_T)})\Big)$
      end for
    end for
    $L_{\text{sample}} \leftarrow \frac{1}{|y|} L_{\text{sample}}$
    $L_{\text{batch}} \leftarrow L_{\text{batch}} + L_{\text{sample}}$
  end for
  Update student: $\theta \leftarrow \theta - \eta \nabla_\theta L_{\text{batch}}$
  (Optional) Update teacher via EMA: $\theta' \leftarrow \tau\theta' + (1-\tau)\theta$
end for

5 Planned Experiments

5.1 Methods / Ablations

- **Compression levels:** fix teacher R (e.g. $R = 4$) and vary student $\tilde{R} \in \{2, 3, 4\}$.
- **Cross-loop vs. terminal-only:** Eq 1 vs. terminal-only compression baseline ($\alpha_{\tilde{R}} = 1$, $\phi(\tilde{R}) = R$).
- **Mapping functions:** shift mapping (e.g $1 \to 3$, $2 \to 4$) vs. linear mapping (e.g $1 \to 2$, $2 \to 4$) vs. fixed targets (e.g $1 \to 4$, $2 \to 4$).
- **Weight schedules:** uniform vs. late-heavy vs. adaptive over student loops.
- **Divergences:** KL vs. reverse-KL vs. JSD.
- **Teacher variants:** OPSD-style ($c = a$) vs. SDPO-style ($c = f$).

5.2 Benchmarks

- **Math:** Math500, BeyondAIME, AMO-Bench, HMMT25
- **Coding:** LiveCodeBench, HumanEval Pro, Codeforces, etc.

5.3 Metrics

- **Accuracy:** Pass@1 and Pass@k ($k \in \{2, 4, 8, 16, 32, 64\}$), average@16, reported at fixed inference loop budgets $\tilde{R}$.
- **Compute–quality tradeoff:** accuracy vs. inference loops, comparing teacher (R loops) to compressed student ($\tilde{R}$ loops).
- **Efficiency:** wall-clock latency / tokens-per-second and effective speedup from using $\tilde{R} < R$.
- **Uncertainty:** entropy of per-loop distributions (student across $r \leq \tilde{R}$; teacher across $r \leq R$).

6 Implementation Notes

- **Logit storage:** cross-loop distillation requires teacher logits at loops $\{\phi(r)\}_{r=1}^{\tilde{R}}$ (and student logits for $r \leq \tilde{R}$). To reduce memory, consider storing only the mapped teacher loops, or using top-K / chunked KL.
- **Teacher stabilization:** EMA or frozen teacher can help stabilize training when targeting sharp, late-loop teacher distributions.
- **Training compute:** the teacher runs R loops with privileged conditioning; the student runs $\tilde{R}$ loops. Overall training cost increases, but inference cost decreases when deploying the compressed student.

7 Pros & Cons

Cons:

- May not outperform baselines of RLTT/GRPO/SFT.
- Gains may depend on careful choices of ϕ and $\{\alpha_r\}$, and could be benchmark-sensitive.

Pros:

- Maintains on-policy, dense token-level learning signal while shifting competence earlier in latent computation time.
- Potentially reduces inference cost/latency while preserving reasoning quality.

References

[1] Anonymous et al. Looping language models. *arXiv preprint*, 2025.

[2] Anonymous et al. Prioritize the process: Trajectory training for better reasoning in rlvr. *arXiv preprint*, 2025.

[3] Jonas Hübotter et al. Reinforcement learning via self-distillation (sdpo). *arXiv preprint*, 2026.

[4] Siyan Zhao et al. Self-distilled reasoner: On-policy self-distillation for large language models. *arXiv preprint*, 2026.