LoopRPT: Reinforcement Pre-Training for Looped Language Models

Guo Tang[1*] **Shixin Jiang**[1*] **Heng Chang**[3*] **Zihan Zhang**[4] **Nuo Chen**[5]
Yuhan Li[5] **Huiming Fan**[1] **Jia Li**[6] **Ming Liu**[1,2] **Bing Qin**[1,2†]
[1]Harbin Institute of Technology, Harbin, China [2]Peng Cheng Laboratory, Shenzhen, China
[3]Tsinghua University [4]Harvard University [5]HY Foundation Model Team, Tencent
[6]Hong Kong University of Science and Technology (Guangzhou)
{gtang,sxjiang,mliu,qinb}@ir.hit.edu.cn changh17@tsinghua.org.cn

Abstract

Looped language models (LoopLMs) perform iterative latent computation to refine internal representations, offering an alternative to explicit chain-of-thought (CoT) reasoning. However, existing reinforcement learning (RL) paradigms primarily target output tokens, creating a structural mismatch with looped architectures whose reasoning unfolds implicitly. In this work, we propose **LoopRPT**, a reinforcement pre-training framework tailored for LoopLMs. By reframing next-token prediction as a next-token reasoning task, LoopRPT assigns reinforcement signals directly to latent steps using an EMA teacher reference and noisy latent rollouts. This formulation enables RL to directly shape intermediate representations, compressing effective reasoning into fewer iterations. We instantiate LoopRPT on Ouro across multiple model scales. Results show that LoopRPT improves per-step representation quality and achieves Pareto dominance in accuracy–computation trade-offs. Large gains on hard tokens indicate that LoopRPT enhances early-stage reasoning rather than merely encouraging premature exits. Our findings highlight reinforcement pre-training as a principled paradigm for learning efficient latent reasoning in LoopLMs.

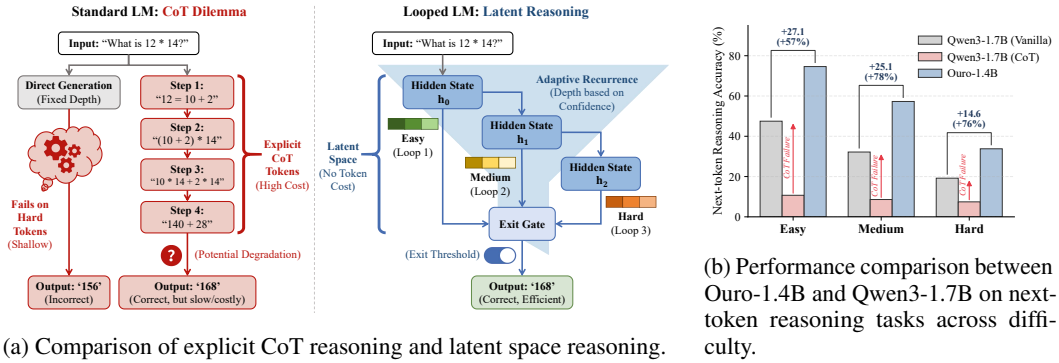
1 Introduction

Modern Large Language Models (LLMs) are trained to "think" primarily through explicit text generation, such as chain-of-thought (CoT) prompting [Wei et al., 2022]. While effective, this paradigm defers reasoning to post-training and often underuses the structural information available in pre-training data [Christiano et al., 2017, Wen et al., 2025, Guo et al., 2025]. In contrast, Looped Language Models (LoopLMs), such as Ouro, use a parameter-shared looped backbone where internal representations are refined recurrently in latent space [Zhu et al., 2025]. This enables iterative computation and deeper reasoning without a proportional increase in parameter count, compressing multi-step reasoning into latent transitions rather than explicit tokens (Fig. 1a).

However, effectively incentivizing these looped structures using reinforcement learning (RL) remains a challenge. Traditional Reinforcement Learning with Verifiable Rewards (RLVR) paradigms primarily operate on output tokens, creating a structural mismatch with looped architectures whose reasoning unfolds implicitly [Shao et al., 2024, Wen et al., 2025]. As identified in Ouro [Zhu et al., 2025], the dynamic early-exit mechanism, which adaptively sets computation depth, introduces instability and credit-assignment issues for RLVR alignment. This leads to a fundamental research question:

[*]Equal contribution.
[†]Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).



(a) Comparison of explicit CoT reasoning and latent space reasoning.

(b) Performance comparison between Ouro-1.4B and Qwen3-1.7B on next-token reasoning tasks across difficulty.

Figure 1: **Motivation for LoopRPT. (a)** Standard LLMs reason through explicit chain-of-thought tokens, whereas LoopLMs compress multi-step reasoning into latent space. **(b)** LoopLMs achieve higher next-token-reasoning accuracy than standard LLMs across difficulty levels.

How to incentivize LoopLMs using RL with performance gains compared to non-recursive transformer models?

Current RLVR tasks rely on sparse rewards derived from final output tokens, which are insufficient for the dense, multi-step latent reasoning required by LoopLMs. Recently, **Reinforcement Pre-Training (RPT)** has emerged as a new scaling paradigm, reframing next-token prediction as a reasoning task [Hatamizadeh et al., 2025, Dong et al., 2025]. By treating internal computation as an exploratory action and rewarding the predictive gain for future tokens, RPT turns vast text data into dense, self-supervised RL signals, a path toward training looped architectures.

In this work, we make the first attempt to unify looped architectures with the RPT paradigm. Our initial analysis (Fig. 1b) shows that LoopLMs are naturally "reasoning-friendly" for next-token tasks: vanilla LLMs, lacking training for CoT-based next-token reasoning, degrade sharply, while looped models stay accurate across difficulty levels. This suggests that the knowledge-manipulation strength of LoopLMs can be unlocked by an RL algorithm designed for latent recurrence.

To fill this gap, we propose **LoopRPT**, a **R**einforcement **P**re-**T**raining Framework tailored for **Loop**ed LMs. Built upon the next-token reasoning task, LoopRPT assigns reinforcement signals directly to latent reasoning steps. The framework incorporates three key innovations: (i) an **entropy-based selector** that identifies "hard tokens" where reasoning is most beneficial [Wang et al., 2025] , (ii) **step-wise rewards** computed against a dynamic EMA teacher to shape intermediate representations [Tarvainen and Valpola, 2017, Hatamizadeh et al., 2025] , and (iii) **noisy latent rollouts** to jointly optimize the exit policy and backbone representations. By combining accuracy gains with a difficulty-aware time penalty, LoopRPT encourages early effective exits without sacrificing correctness on challenging tokens.

We conduct experiments on Ouro-1.4B and Ouro-2.6B using the OMNI-MATH [Gao et al., 2024a] and general reasoning datasets. LoopRPT improves per-step representation quality and achieves a better accuracy-computation trade-off: fewer average steps while maintaining or improving hard-token accuracy. Gains transfer to downstream benchmarks, with improvements in GSM8K [Cobbe et al., 2021], MBPP [Austin et al., 2021], and HumanEval [Chen et al., 2021].

Our primary contributions are summarized as follows:

- We introduce **LoopRPT**, the first framework to apply the Reinforcement Pre-Training paradigm to Looped LMs, addressing the mismatch between latent reasoning and sparse token-level rewards.
- We design a **step-wise reinforcement objective** using an EMA teacher and noisy latent rollouts to provide dense supervision over internal recurrent transitions.
- Empirical results show that LoopRPT achieves Pareto dominance in accuracy-computation trade-offs across multiple scales, outperforming both vanilla looped models and standard LLMs with explicit CoT.

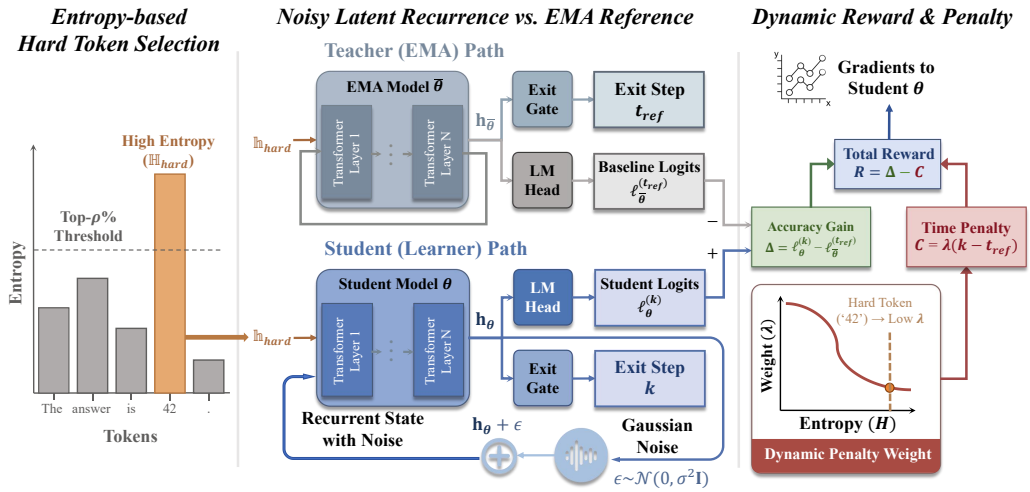


Figure 2: **LoopRPT** assigns reinforcement signals directly to latent reasoning steps. An entropy-based selector identifies hard tokens, and a student trains against an EMA teacher via step-wise rewards that combine an accuracy gain over the teacher with a dynamic time penalty, shaping intermediate representations and encouraging earlier effective exits.

2 Preliminaries

Looped Language Models (LoopLMs). Unlike standard transformers, LoopLMs refine internal representations through iterative latent computation. Given context $x_{<t}$, the model updates a latent state $\mathbf{h}^{(k)}$ via a parameter-shared backbone f_θ:

$$\mathbf{h}^{(k+1)} = f_\theta(\mathbf{h}^{(k)}, x_{<t}), \quad k = 0, \ldots, K-1, \tag{1}$$

The next-token distribution is then derived from the final state: $p_\theta(x_t \mid x_{<t}) = g_\theta(\mathbf{h}^{(K)})$, where g_θ represents an LM head. This enables multi-step reasoning in latent space without emitting intermediate tokens.

Standard Next-Token Prediction (NTP) remains the dominant objective, maximizing the log-likelihood: $\mathcal{L}_{\text{NTP}}(\theta) = -\sum \log p_\theta(x_t \mid x_{<t})$. However, NTP treats predictions independently and does not model the latent reasoning required for difficult tokens.

Reinforcement Pre-Training (RPT) reframes prediction as a reasoning task. The model samples an output $o_t \sim \pi_\theta$ and receives a verifiable reward $r(o_t, x_t)$ based on the ground-truth continuation. The objective maximizes the expected reward over the corpus: $\mathcal{J}_{\text{RPT}}(\theta) = \mathbb{E}_{x_{<t}} \mathbb{E}_{o_t \sim \pi_\theta(\cdot|x_{<t})}\big[r(o_t, x_t)\big]$. Using intrinsic corpus rewards, RPT scales RL to large datasets and incentivizes next-token reasoning over simple likelihood maximization.

3 Method

In the RLVR paradigm, rewards are sparse, derived solely from final answer tokens. For looped LMs that reason over multiple latent steps, this sparsity exacerbates credit assignment across intermediate iterations. We introduce dense step-wise reward signals on challenging next-token predictions, enabling looped LMs to improve intermediate reasoning and learn an adaptive exit policy.

LoopRPT is a reinforcement pre-training framework for looped LMs, where computation unfolds over latent steps and an exit mechanism sets the number of recurrent iterations per token. As illustrated in Fig. 2, LoopRPT (i) selects *hard* next-token prediction instances via entropy, (ii) defines a *step-wise* reward on latent iterations relative to an EMA teacher, and (iii) jointly optimizes the exit policy and intermediate representations using noisy latent rollouts and step-weighted next-token learning. [1]

[1] Full training algorithms are provided in App. I.

3.1 Looped LM with an exit-step distribution

As in Sec. 2, each latent step produces next-token logits and a scalar exit-gate logit $a^{(k)}$, with exit probability $\lambda^{(k)} = \sigma(a^{(k)})$. Following Ouro's early-exit mechanism [Zhu et al., 2025], we define a survival probability $s^{(1)} = 1$ and update

$$\pi_\theta(k) = \begin{cases} \lambda^{(k)} s^{(k)}, & k < K, \\ s^{(K)}, & k = K, \end{cases} \tag{2}$$

$$s^{(k+1)} = s^{(k)}\big(1 - \lambda^{(k)}\big) \ \ (k < K), \tag{3}$$

where the last step absorbs the remaining probability mass, ensuring $\sum_{k=1}^{K} \pi_\theta(k) = 1$ exactly. We compute the CDF $\Pi(k) = \sum_{j \le k} \pi_\theta(j)$ and determine the threshold exit step $t(\tau) = \min\{k : \Pi(k) \ge \tau\}$. For numerical robustness, if no step satisfies $\Pi(k) \ge \tau$ (e.g., due to finite precision), we set $t(\tau) = K$.

3.2 Entropy-based hard-token selection

Applying reinforcement learning uniformly to all tokens is inefficient because most tokens are easy and contribute weak learning signals. LoopRPT therefore trains primarily on *hard* next-token instances identified by teacher uncertainty.[2] Concretely, for each position t in a sequence we compute the teacher's next-token distribution and entropy

$$H_t = -\sum_{v} p_{\bar\theta}(v \mid x_{<t}) \log p_{\bar\theta}(v \mid x_{<t}), \tag{4}$$

and select a top fraction of positions (top-$\rho\%$) within each example as hard tokens.[3] All losses in Sec. 3.3–3.4 are applied only on the selected positions.

3.3 Step-wise rewards with an EMA teacher

In order to provide a dense credit assignment per-step anchored by a stable moving reference [Hatamizadeh et al., 2025], LoopRPT assigns reinforcement signals *directly to latent reasoning steps* and maintains an EMA teacher $\bar\theta$. $\bar\theta$ tracks student parameters θ and serves as a dynamic reference for both exiting and reward baselines.

Teacher reference step. For each token, we compute the teacher's exit distribution $\pi_{\bar\theta}$ and define the teacher reference step $t_{\text{ref}} = \min\Big\{k : \sum_{j \le k} \pi_{\bar\theta}(j) \ge \tau\Big\}$. Intuitively, t_{ref} is the teacher's notion of sufficient computation at the current exit threshold.

Accuracy gain. Let $\ell_\theta^{(k)} = \log p_\theta(x_t \mid x_{<t}; \mathbf{h}^{(k)}) \in \mathbb{R}^{B\times(S-1)}$ denote the student log-probability of the ground-truth next token when read out from step k. We take the teacher log-probability at the reference step as a per-token baseline,

$$b_{\text{ref}} = \ell_{\bar\theta}^{(t_{\text{ref}})} \in \mathbb{R}^{B\times(S-1)}. \tag{5}$$

The step-wise accuracy gain is

$$\Delta_{\text{acc}}(k) = \ell_\theta^{(k)} - b_{\text{ref}}. \tag{6}$$

Difficulty-aware time penalty. To encourage earlier *effective* exits without forcing premature termination on difficult tokens, we penalize excessive computation relative to the teacher reference step:

$$C(k) = \lambda_t\,(k - t_{\text{ref}}). \tag{7}$$

Here λ_t is a token-dependent penalty weight derived from teacher uncertainty at the reference step. Specifically, we compute the entropy H_t^{ref} at the teacher's reference step distribution, normalize it by $\log|\mathcal{V}|$ and clamp it to obtain a difficulty score $d_t = \text{Clamp}\big(H_t^{\text{ref}}/\log|\mathcal{V}|,\, 0,\, 1\big)$. We introduce the base coefficient λ_{base} and the scaling coefficient λ_{scale}, thus λ_t is defined as

$$\lambda_t = \lambda_{\text{base}}\Big(1 + \lambda_{\text{scale}}(1 - d_t)\Big), \tag{8}$$

[2] We provide theoretical analysis in App. B.1.

[3] Details are deferred to App. C.

so that easier tokens (lower entropy) incur larger time penalties, while harder tokens are penalized less.

Total step-wise reward. Combining the above, LoopRPT defines a reward for each latent step:

$$R(k) = \Delta_{\text{acc}}(k) - C(k). \tag{9}$$

This reward provides dense supervision over latent iterations, shaping intermediate representations and aligning them with early exiting (Fig. 2).

Finally, we compute a per-token *step advantage* by normalizing $R(k)$ across steps $k = 1, \ldots, K$:

$$\widehat{A}(k) = \frac{R(k) - \mu_R}{\sigma_R + \epsilon}, \quad \mu_R = \frac{1}{K}\sum_{j=1}^{K} R(j), \quad \sigma_R = \sqrt{\frac{1}{K}\sum_{j=1}^{K}(R(j) - \mu_R)^2}, \tag{10}$$

which will be used to emphasize beneficial latent steps in the representation learning objective.

3.4 Joint optimization via noisy latent rollouts

LoopRPT jointly trains (i) the *exit policy* induced by the gate distribution π_θ and (ii) the *backbone* representations $\{\mathbf{h}^{(k)}\}$ that support accurate prediction at early steps.

Noisy latent rollouts for the exit policy. As LoopLM reasoning is implicit in latent space, we obtain on-policy variability by injecting Gaussian noise into the recurrent hidden states.[4] For each selected token, we draw G rollouts and perturb the latent states during recurrence:

$$\mathbf{h}^{(k)} \leftarrow \mathbf{h}^{(k)} + \boldsymbol{\epsilon}^{(k)}, \quad \boldsymbol{\epsilon}^{(k)} \sim \mathcal{N}(0, \sigma^2 \mathbf{I}), \tag{11}$$

yielding a rollout-specific exit distribution $\pi_\theta^{(g)}(\cdot)$. We then sample an exit step $t^{(g)} \sim \pi_\theta^{(g)}$ and define the rollout reward by indexing the step-wise reward table in Eq. (9), $r^{(g)} = R\big(t^{(g)}\big)$. To stabilize policy learning, we use group-wise normalization across the G rollouts for the same input token:

$$A^{(g)} = \frac{r^{(g)} - \text{mean}_g[r^{(g)}]}{\text{std}_g[r^{(g)}] + \epsilon}. \tag{12}$$

The policy-gradient loss is

$$\mathcal{L}_{\text{PG}} = -\mathbb{E}_g\big[A^{(g)} \log \pi_\theta^{(g)}(t^{(g)})\big], \tag{13}$$

applied only on the selected hard-token positions.

Step-weighted representation learning. LoopRPT also strengthens intermediate representations so that early latent steps can predict the correct token. Using a deterministic (noise-free) forward pass, we compute per-step log-probabilities $\ell_\theta^{(k)}$ and the (noise-free) exit distribution $\pi_\theta(k)$. We then optimize a step-weighted next-token objective:

$$\mathcal{L}_{\text{rep}} = -\sum_{k=1}^{K} w_k \, \ell_\theta^{(k)}, \tag{14}$$

$$w_k = \text{softmax}\big[\pi_\theta(k)\big(1 + \text{ReLU}(\widehat{A}(k))\big)\big]. \tag{15}$$

The $\pi_\theta(k)$ term emphasizes likely exit steps; the advantage-shaping term focuses learning on higher-reward steps under Eq. (9).

Regularization and total objective. We add an entropy bonus on the exit distribution to prevent early collapse, $\mathcal{L}_{\text{ent}} = -\mathbb{E}\big[\sum_k \pi_\theta(k) \log(\pi_\theta(k))\big]$, and a KL-style trust-region penalty to the EMA teacher based on a token-level surrogate computed from teacher and student step-wise log-probabilities.[5] The final training loss is

$$\mathcal{L} = \alpha \, \mathcal{L}_{\text{PG}} + \beta \, \mathcal{L}_{\text{rep}} + \gamma \, \mathcal{L}_{\text{ent}} + \delta \, \mathcal{L}_{\text{KL}}. \tag{16}$$

After each update to θ, we update the EMA teacher parameters $\bar{\theta}$. [6]

[4] We provide theoretical analysis in App. B.2.

[5] Details are provided in App. D.

[6] We provide theoretical analysis in App. B.3.

4 Experimental Setup

We use OMNI-MATH [Gao et al., 2024a] as the training data. OMNI-MATH contains 4,428 competition-level mathematical problems paired with solutions, among which 200 examples are held out as the validation set. We first conduct experiments on Ouro-1.4B and then scale up to Ouro-2.6B. Both the teacher model and the student model are initialized from the same base model. The teacher parameters are maintained as an exponential moving average of the current student parameters (momentum 0.995), providing a stable reference for generating outputs, while the student model is updated via backpropagation. For token-level updates, we treat the teacher model as the entropy proxy and compute token-level entropy based on its output distribution. We then construct an entropy mask to suppress gradient updates on low-entropy tokens [Wang et al., 2025]. [7]

Language modeling. Following RPT [Dong et al., 2025], we evaluate the performance on the held-out validation set of 200 examples from OMNI-MATH. We bucket tokens by their entropy and report performance on easy/medium/hard splits, defined by thresholds 0.5, 1.0 and 1.5, respectively. For the Ouro models, we adopt the corresponding base model as an entropy proxy and evaluate performance under two inference settings: maximum-loop execution and adaptive early exit. We also evaluate the non-loop Qwen3-1.7B under two settings: the vanilla mode that performs NTP directly, and the CoT mode that first generates a chain-of-thought and then performs NTP.

End tasks. We evaluate zero-shot performance on MMLU [Hendrycks et al., 2021a], MMLU-Pro [Wang et al., 2024], BBH [Suzgun et al., 2023], ARC-C [Clark et al., 2018], HellaSwag [Zellers et al., 2019], Winogrande [Sakaguchi et al., 2021], GSM8K [Cobbe et al., 2021], MBPP [Austin et al., 2021], MBPP+ [Liu et al., 2023], HumanEval Chen et al. [2021], and HumanEval+ [Liu et al., 2023]. Following Ouro [Zhu et al., 2025], all evaluations were conducted using lm-eval-harness [Gao et al., 2024b] and evalplus [Liu et al., 2023]. [8]

Baselines and additional evaluations. To separate the effect of the objective from additional training, we compare LoopRPT against two compute-matched controls that share its checkpoint, data, step budget, and schedule: Continued-NTP (SFT) and GRPO/RLVR. We also evaluate MATH500, the full Hendrycks MATH set, and AIME 2024/2025 for transfer to olympiad-level mathematics, add a no-CoT zero-shot pass over MMLU, GSM8K, and BBH, and repeat the recipe on the broader OpenThoughts-3 corpus. [9]

5 Experimental Results

5.1 Main Results

Next-token Reasoning (NTR). Table 1 shows that LoopRPT consistently improves accuracy and efficiency across all scales. On Ouro-2.6B, LoopRPT boosts Hard Peak accuracy by +3.58 while reducing average steps from 3.51 to 2.28. While explicit CoT prompting on Qwen3-1.7B severely degrades performance, LoopRPT successfully compresses effective computation into fewer latent iterations without sacrificing correctness.

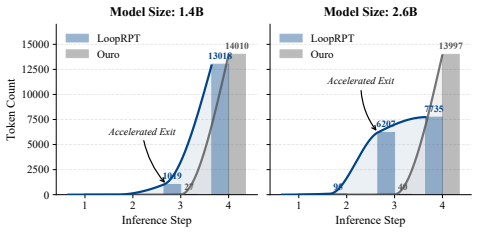


Figure 3: Distribution of exit steps on MMLU. LoopRPT shifts tokens to earlier iterations while keeping final-step exits dominant.

Downstream Benchmarks. As summarized in Tables 2 and 12, LoopRPT delivers consistent gains over Ouro, particularly in coding (MBPP+ +2.91) and mathematical reasoning. At the 2.6B scale, GSM8K accuracy increases from 81.76 to 85.36, alongside improvements in HumanEval+, indicating that reinforcement pre-training on hard instances effectively transfers to complex program synthesis.

Exit Dynamics. LoopRPT strengthens adaptive exiting by increasing early-step exits while maintaining final-step dominance (Fig. 3). At larger scales, the mechanism becomes more aggressive,

[7] Full hyperparameters and training details are deferred to Appendix E (Table 6).

[8] Detailed evaluation settings and metrics are provided in Appendix E (Table 7).

[9] The full compute-matched matrix, the additional evaluation protocols, and the broader-corpus study are given in Appendices F.1 and F.3.

Table 1: Performance comparison on next-token reasoning task across three difficulty levels. *Peak* indicates reasoning up to the maximum latent loops ($K = 4$), while *Adap.* indicates adaptive early exiting. Subscripts indicate improvement ($+$) or degradation ($-$) compared to baseline.

Model	Method	Easy		Medium		Hard	
		Acc (↑)	Avg step (↓)	Acc (↑)	Avg step (↓)	Acc (↑)	Avg step (↓)
Qwen3-1.7B	Vanilla	47.49	–	32.18	–	19.19	–
	+CoT	$10.70_{-36.79}$	–	$8.60_{-23.58}$	–	$7.44_{-11.75}$	–
Ouro-1.4B	Peak	74.62	4.00	57.28	4.00	33.79	4.00
	+LoopRPT	**75.38**$_{+0.76}$	4.00	**58.29**$_{+1.01}$	4.00	**34.74**$_{+0.95}$	4.00
	Adap.	74.40	3.34	57.20	3.53	33.91	3.75
	+LoopRPT	**75.00**$_{+0.60}$	**2.50**$_{-0.84}$	**57.72**$_{+0.52}$	**2.81**$_{-0.72}$	**34.82**$_{+0.91}$	**3.07**$_{-0.68}$
Ouro-2.6B	Peak	74.33	4.00	57.19	4.00	34.52	4.00
	+LoopRPT	**76.89**$_{+2.56}$	4.00	**61.15**$_{+3.96}$	4.00	**38.10**$_{+3.58}$	4.00
	Adap.	74.51	3.24	57.35	3.35	34.35	3.51
	+LoopRPT	**76.07**$_{+1.56}$	**2.05**$_{-1.19}$	**60.21**$_{+2.86}$	**2.18**$_{-1.17}$	**37.24**$_{+2.89}$	**2.28**$_{-1.23}$

Table 2: Comparison of 1.4B LoopLM model with 1-4B parameter baselines. Results for vanilla LLMs are taken from Zhu et al. [2025], with Ouro results obtained via our reimplementation. The best score is **bolded**, and the second-best is underlined. Subscripts indicate LoopRPT's improvement over Ouro baseline.

Benchmark	Gemma3 1B	Llama3.2 1.2B	Qwen2.5 1.5B	Qwen3 1.7B	Qwen2.5 3B	Llama3.2 3B	Qwen3 4B	Gemma3 4B	Ouro 1.4B	**LoopRPT 1.4B**
General Tasks										
MMLU	39.85	45.46	60.99	62.46	65.62	59.69	**73.19**	58.37	67.35	67.62$_{+0.27}$
MMLU-Pro	11.31	11.80	29.11	37.27	37.87	33.34	**51.40**	34.61	48.64	49.21$_{+0.57}$
BBH	30.26	30.72	43.66	53.51	55.37	39.45	70.95	66.32	70.97	**71.37**$_{+0.40}$
ARC-C	39.25	41.98	54.44	55.72	55.46	52.47	**63.65**	60.92	60.95	61.10$_{+0.15}$
HellaSwag	56.12	59.35	67.73	67.09	74.54	73.09	**75.66**	75.58	74.30	74.93$_{+0.63}$
Winogrande	58.72	62.75	66.77	66.30	70.17	69.14	71.19	71.07	71.67	**71.98**$_{+0.31}$
Math & Coding Tasks										
GSM8K	2.05	7.05	60.73	70.28	74.60	67.20	72.86	68.69	78.17	**78.92**$_{+0.75}$
HumanEval	6.70	19.50	52.40	66.50	68.90	29.90	**77.40**	34.80	69.47	69.51$_{+0.04}$
HumanEval+	5.50	17.40	46.30	59.80	62.20	26.20	**70.70**	29.30	67.01	67.07$_{+0.06}$
MBPP	12.40	35.70	60.30	68.00	63.00	50.30	**78.80**	60.60	73.02	75.40$_{+2.38}$
MBPP+	10.10	29.10	50.00	58.50	54.20	39.70	**65.90**	51.10	60.85	63.76$_{+2.91}$

with a shift towards step-3 exits. This reflects improved confidence calibration and earlier latent convergence, aligning with the accuracy–compute Pareto dominance observed in NTR.

5.2 Compute-Matched Controls

Comparing LoopRPT against the original Ouro checkpoint alone cannot separate the reinforcement objective from additional training on OMNI-MATH. We add two control arms that share its checkpoint, data, step budget, and schedule, changing only the objective: *Continued-NTP* uses plain next-token prediction, and *GRPO/RLVR* uses group-relative policy optimization on output-token rewards. Table 3 reports both at both scales. Continued-NTP stays close to the baseline on every metric except MATH500, so a further pass over the corpus does not reproduce the gains. GRPO fails to

Table 3: **Compute-matched controls.** All arms share the checkpoint, data, step budget, and schedule; only the objective changes. NTR-Hard is hard-split accuracy under adaptive early exit; best per scale in **bold**.

Scale	Method	NTR-Hard	GSM8K	MATH500	ARC-C	HellaSwag	BBH	MBPP+
Ouro-1.4B	Base	33.91	78.17	20.60	60.95	74.30	70.97	60.85
	+ Continued-NTP	34.29	77.63	29.40	60.58	74.51	71.10	62.17
	+ GRPO / RLVR	33.00	4.17	12.60	49.23	74.36	28.84	20.58
	+ LoopRPT	**34.82**	**78.92**	**37.00**	**61.10**	**74.93**	**71.37**	**63.76**
Ouro-2.6B	Base	34.35	81.76	43.00	66.12	79.28	77.98	64.81
	+ Continued-NTP	35.21	82.41	43.20	66.75	79.74	78.19	63.76
	+ GRPO / RLVR	33.00	6.63	18.00	49.89	78.07	29.36	57.58
	+ LoopRPT	**37.24**	**85.36**	**43.50**	**66.89**	**80.03**	**78.24**	**65.08**

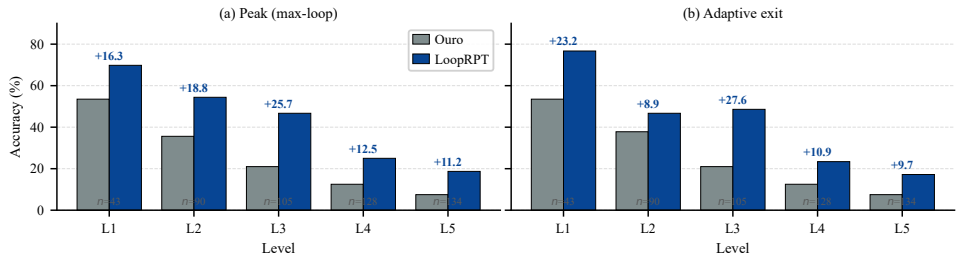


Figure 4: **MATH500 accuracy by difficulty level** (Ouro-1.4B vs. LoopRPT-1.4B). Levels use the official Hendrycks problem-level labels, which are independent of the token-level entropy that drives token selection during training. LoopRPT improves every level under both maximum-loop (a) and adaptive early exit (b); the per-level sample sizes are shown at the bottom of each panel.

Table 4: **Transfer to hard mathematics.** Subscripts give the change over Ouro.

Scale	Benchmark	Ouro	LoopRPT
1.4B	MATH500	20.60	37.00$_{+16.40}$
	MATH (full)	71.74	76.20$_{+4.46}$
	AIME 2024	6.67	6.67
	AIME 2025	6.67	10.00$_{+3.33}$
	GSM8K	78.17	78.92$_{+0.75}$
2.6B	MATH500	43.00	43.50$_{+0.50}$
	MATH (full)	76.50	77.50$_{+1.00}$
	GSM8K	81.76	85.36$_{+3.60}$

Table 5: Ablation study of LoopRPT-1.4B. *Overall* is the average of the three difficulty levels.

Varies	Hard			Overall		
	Peak	Adap.	Avg Step	Peak	Adap.	Avg Step
LoopRPT	**34.74**	**34.82**	**3.07**	**56.14**	**55.85**	**2.79**
w/o Gaussian Noise	34.69	34.76	3.29$_{(\uparrow)}$	55.86	55.57	3.17$_{(\uparrow)}$
w/o $\mathcal{L}_{\text{PG}} + \mathcal{L}_{\text{KL}}$	34.59	34.17	3.28$_{(\uparrow)}$	55.89	55.51	3.14$_{(\uparrow)}$
w/o $\mathcal{L}_{\text{rep}} + \mathcal{L}_{\text{ent}}$	34.02	33.75	3.46$_{(\uparrow)}$	55.52	55.19	3.24$_{(\uparrow)}$
w/o Token Selection	34.60	34.52	2.89$_{(\downarrow)}$	55.76	55.41	2.49$_{(\downarrow)}$
w/o Time Penalty	34.57	34.80	3.30$_{(\uparrow)}$	55.94	55.78	3.19$_{(\uparrow)}$

converge: GSM8K falls to a few percent, BBH drops by over 40 points, and ARC-C and MBPP+ degrade, matching the credit-assignment difficulty of sparse output rewards over latent iterations (Sec. 3). LoopRPT keeps the same budget and raises every metric in the table, and its margin over Continued-NTP isolates the objective itself.

5.3 Transfer to Hard Mathematics and No-CoT Reasoning

Hard mathematics. The original suite contains one mathematics benchmark, GSM8K, which Ouro-2.6B nearly saturates at 81.76. We add MATH500, the full 5,000-problem Hendrycks MATH set, and AIME 2024/2025, all scored with the same protocol for Ouro and LoopRPT. Table 4 shows that the gain extends beyond the benchmark closest to the training distribution: MATH500 and the broader Hendrycks set both improve at each scale, and AIME 2025 improves as well. AIME 2024 (30 problems) is the only benchmark that stays flat, and it is the furthest from the training distribution.

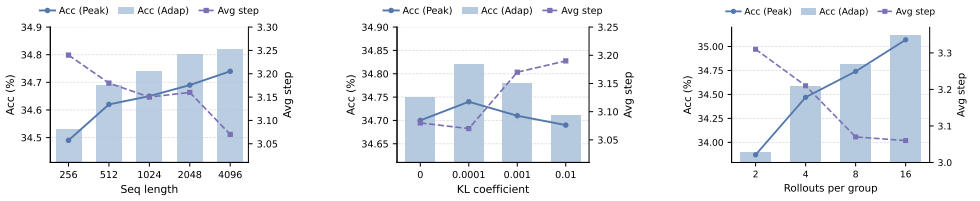
Difficulty stratification. Fig. 4 breaks MATH500 down by the official Hendrycks problem-level labels (Level 1–5), which are assigned per problem and are independent of the training-time token entropy. LoopRPT improves every level by 9 to 28 points across the two inference settings, including the hardest level.

No-CoT zero-shot. GSM8K and BBH use few-shot chain-of-thought in the original suite, so we also run a no-CoT zero-shot pass (few-shot $= 0$, GSM8K answers directly, BBH uses the no-CoT template; Table 10 in the appendix). No-CoT accuracy is far below the few-shot numbers on both, so explicit CoT overstates a looped model's internal reasoning. The gain survives the removal: LoopRPT raises no-CoT GSM8K by 13.7 points at 1.4B and 26.0 at 2.6B, its largest downstream gain in any setting. MMLU, unrelated to the training signal, barely changes.

5.4 Ablation Study

Table 5 ablates key design choices of LoopRPT on Ouro-1.4B across *Hard* and *Overall* splits. Overall, removing any component degrades accuracy or efficiency, confirming that LoopRPT relies on multiple complementary ingredients.

Stabilization and trajectory diversity. Disabling Gaussian noise reduces accuracy and increases average exit steps, suggesting that stochastic latent rollouts facilitate exploration and earlier latent



(a) Effect of the maximum sequence length.

(b) Effect of the KL regularization coefficient.

(c) Effect of the group size during noisy latent rollouts.

Figure 5: Sensitivity analysis of LoopRPT hyperparameters on the next-token reasoning task. All results are reported on the *hard* split, showing accuracy and average exit steps under different settings.

convergence. Similarly, removing policy-gradient and KL stabilizers leads to large performance drops and increased latency under adaptive inference, highlighting their importance for stable on-policy learning and calibrated early exiting.

Representation shaping and efficiency. Removing $\mathcal{L}_{\text{rep}} + \mathcal{L}_{\text{ent}}$ yields the largest degradation in both accuracy and efficiency, underscoring that representation regularization and entropy shaping are vital for learning effective latent transitions. While disabling token selection slightly reduces steps, it notably hurts Hard-split accuracy, proving that hard-token updates preferentially allocate capacity to difficult reasoning tasks. Finally, removing the time penalty modestly hurts accuracy while consistently increasing average steps, confirming its role in explicitly incentivizing early exit and improving the accuracy–compute trade-off.

6 Analysis and Discussion

6.1 Hyperparameter Sensitivity

LoopRPT exhibits low sensitivity to key hyperparameters and follows a predictable accuracy–compute trade-off. Increasing the maximum sequence length (see Fig. 5a) consistently improves both *Peak* and *Adap.* accuracy, while the average exit step slightly decreases or saturates, suggesting that longer context helps latent computation converge earlier. The KL regularization coefficient (Fig. 5b) shows a mild sweet spot: moderate KL improves *Adap.* performance, whereas overly strong KL tends to reduce gains and yields more conservative exiting (higher average steps), consistent with over-constraining on-policy updates. Finally, enlarging the GRPO group size (Fig. 5c) improves accuracy and reduces average steps, indicating that better group-wise advantage estimates strengthen both learning signal and early-exit calibration, with diminishing returns at larger group sizes.

6.2 Analysis of Latent Reasoning

To better understand how LoopRPT improves the accuracy–computation trade-off, we analyze the latent reasoning dynamics of LoopLMs at the level of intermediate computation steps. Fig. 6 presents the accuracy–computation Pareto curves under varying exit thresholds. Across three difficulty buckets, LoopRPT consistently achieves higher accuracy with fewer average computation steps compared to the baseline Ouro model. Notably, this Pareto dominance holds across model scales, indicating that the gains introduced by LoopRPT are not tied to a specific parameter regime.

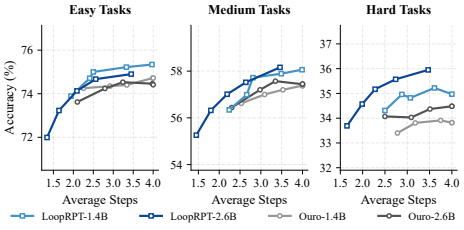


Figure 6: Accuracy–computation trade-offs across difficulty buckets. LoopRPT enhances next-token reasoning accuracy with reduced loop cycles.

To uncover the mechanism behind this improvement, Fig. 7 reports per-step next-token prediction accuracy obtained from intermediate hidden states within a single forward pass. We observe that LoopRPT improves prediction accuracy at every latent step, with particularly pronounced gains at early steps. This effect is most evident for hard tokens, where LoopRPT significantly boosts step-1 and step-2 accuracy.

These results suggest that LoopRPT does not merely encourage earlier exiting, but fundamentally enhances the quality of intermediate representations. By shaping latent reasoning through reinforcement pre-training, LoopRPT compresses effective computation into fewer steps while preserving or improving predictive performance.

7 Related Work

Reinforcement learning for (pre-)training LMs. Modern RL-based post-training for LMs is often instantiated as RL from preference feedback, where a reward model or implicit reward is optimized with policy-gradient updates such as TRPO/PPO [Schulman et al., 2015, 2017] and human preference supervision [Christiano et al., 2017, Stiennon et al., 2020, Ouyang et al., 2022]. Subsequent work reduces reliance on human labels by leveraging AI feedback and rule-based supervision, e.g., Constitutional AI / RLAIF [Bai et al., 2022], and broadens the toolkit beyond reward-model RLHF via direct/implicit preference optimization [Rafailov et al., 2023, Ethayarajh et al., 2024, Hong et al., 2024] and its extensions [Xiao et al., 2025, Jiang et al., 2025, Cen et al., 2025]. In parallel, RLVR exploit programmatic or checker-based supervision to improve reasoning without a learned reward model [Wen et al., 2025, Guo et al., 2025, Shao et al., 2024], but are typically applied as task-specific post-training due to limited availability of verifiable supervision at scale. RPT reframes next-token prediction as a verifiable RL objective, deriving RL signals directly from the pre-training corpus [Dong et al., 2025, Hatamizadeh et al., 2025]. Our work follows this direction and addresses a key bottleneck: token-level feedback is sparse and dominated by easy transitions. By concentrating dense updates on hard next-token instances [Wang et al., 2025], we improve both token-level reasoning and downstream transfer while preserving stable training dynamics.

Looped language models and adaptive depth. Recurrence and adaptive computation have a long history in sequence models. Early approaches introduce dynamic halting and step allocation, such as ACT [Graves, 2016] and Universal Transformers [Dehghani et al., 2019], later extended with sparse/shared-depth variants [Tan et al., 2023] and learned pondering mechanisms [Banino et al., 2021]. Complementary to within-token recurrence, segment recurrence and explicit memory target long-context modeling, e.g., Transformer-XL [Dai et al., 2019], Compressive Transformers [Rae et al., 2019] and Recurrent Memory Transformers [Bulatov et al., 2022]. Another related line is dynamic-depth inference via early exiting or layer-wise halting [Xin et al., 2020, Zhou et al., 2020, Xu and McAuley, 2023, Bajpai and Hanawal, 2025], and recent work allocates compute non-uniformly across tokens and layers [Raposo et al., 2024, Chen et al., 2025]. Closest to our setting are Looped/Depth-recurrent LMs that explicitly build iterative latent computation into the model, enabling adaptive early exit at test time [Zhu et al., 2025, Wu et al., 2025, Li et al., 2025, McLeish et al., 2025], as well as related latent-pondering pretraining schemes [Zeng et al., 2025]. Concurrently, RLTT [Williams and Tureci, 2026] also explores distributing rewards across latent reasoning trajectories in looped architectures. However, RLTT is explicitly designed for task-specific post-training (RLVR) relying on final-answer correctness. Our work sits at this intersection: we use LoopLMs as the substrate and contribute an RL pre-training recipe that strengthens token-level reasoning and calibrates early-exit behavior, addressing the instability and weak supervision that arise when scaling latent recurrence with sparse token-level rewards.

8 Conclusion

We introduced LoopRPT, a reinforcement pre-training framework for looped language models that addresses the inherent sparsity of token-level feedback. LoopRPT concentrates on learning signals on hard next-token instances and provides step-wise supervision over latent recurrence. Experiments demonstrate that LoopRPT improves next-token reasoning across difficulty levels under both maximum-loop execution and adaptive early exit, while simultaneously reducing the average inference steps. These gains translate to consistent improvements on a diverse set of end-task benchmarks, particularly in math and code, and are accompanied by better-calibrated early-exit behavior. Together, our results suggest that combining looped computation with hard-token–focused RL pre-training is an effective and scalable path toward stronger intermediate reasoning and more efficient inference. Future work includes extending the approach to larger scales and broader data mixtures, and further improving robustness of early-exit calibration under distribution shift.

Acknowledgements

The research in this article is supported by the National Science Foundation of China (U22B2059, 62276083), Key Research and Development Program of Heilongjiang Province (2024ZX01A05) , Suzhou Municipal Key Technology R&D Project / Suzhou Municipal Key Core Technology Project (SYG2025069).

References

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Xumeng Wen, Zihan Liu, Shun Zheng, Shengyu Ye, Zhirong Wu, Yang Wang, Zhijian Xu, Xiao Liang, Junjie Li, Ziming Miao, Jiang Bian, and Mao Yang. Reinforcement learning with verifiable rewards implicitly incentivizes correct reasoning in base llms, 2025. URL https://arxiv.org/abs/2506.14245.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Honghui Ding, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jingchang Chen, Jingyang Yuan, Jinhao Tu, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaichao You, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645 (8081):633–638, September 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-09422-z. URL http://dx.doi.org/10.1038/s41586-025-09422-z.

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.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Ali Hatamizadeh, Syeda Nahida Akter, Shrimai Prabhumoye, Jan Kautz, Mostofa Patwary, Mohammad Shoeybi, Bryan Catanzaro, and Yejin Choi. Rlp: Reinforcement as a pretraining objective, 2025. URL https://arxiv.org/abs/2510.01265.

Qingxiu Dong, Li Dong, Yao Tang, Tianzhu Ye, Yutao Sun, Zhifang Sui, and Furu Wei. Reinforcement pre-training. *CoRR*, abs/2506.08007, 2025. doi: 10.48550/ARXIV.2506.08007. URL https://doi.org/10.48550/arXiv.2506.08007.

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xionghui Chen, Jianxin Yang, Zhenru Zhang, Yuqiong Liu, An Yang, Andrew Zhao, Yang Yue, Shiji Song, Bowen Yu, Gao Huang, and Junyang Lin. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning, 2025. URL https://arxiv.org/abs/2506.01939.

Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *Advances in neural information processing systems*, 30, 2017.

Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen, Runxin Xu, Zhengyang Tang, Benyou Wang, Daoguang Zan, Shanghaoran Quan, Ge Zhang, Lei Sha, Yichang Zhang, Xuancheng Ren, Tianyu Liu, and Baobao Chang. Omni-math: A universal olympiad level mathematic benchmark for large language models, 2024a. URL https://arxiv.org/abs/2410.07985.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models, 2021. URL https://arxiv.org/abs/2108.07732.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021. URL https://arxiv.org/abs/2107.03374.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding, 2021a. URL https://arxiv.org/abs/2009.03300.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 95266–95290. Curran Associates, Inc., 2024. doi: 10.52202/079017-3018.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and Jason Wei. Challenging BIG-bench tasks and whether chain-of-thought can solve them. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13003–13051, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.824. URL https://aclanthology.org/2023.findings-acl.824/.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge, 2018. URL https://arxiv.org/abs/1803.05457.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence?, 2019. URL https://arxiv.org/abs/1905.07830.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: an adversarial winograd schema challenge at scale. *Commun. ACM*, 64(9):99–106, August 2021. ISSN 0001-0782. doi: 10.1145/3474381. URL https://doi.org/10.1145/3474381.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatGPT really correct? rigorous evaluation of large language models for code generation. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=1qvx610Cu7.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024b. URL https://zenodo.org/records/12608602.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1889–1897, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/schulman15.html.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 3008–3021. Curran Associates, Inc., 2020.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744. Curran Associates, Inc., 2022.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine Olsson, Christopher Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson, Ethan Perez, Jamie Kerr, Jared Mueller, Jeffrey Ladish, Joshua Landau, Kamal Ndousse, Kamile Lukosuite, Liane Lovitt, Michael Sellitto, Nelson Elhage, Nicholas Schiefer, Noemi Mercado, Nova DasSarma, Robert Lasenby, Robin Larson, Sam Ringer, Scott Johnston, Shauna Kravec, Sheer El Showk, Stanislav Fort, Tamera Lanham, Timothy Telleen-Lawton, Tom Conerly, Tom Henighan, Tristan Hume, Samuel R. Bowman, Zac Hatfield-Dodds, Ben Mann, Dario Amodei, Nicholas Joseph, Sam McCandlish, Tom Brown, and Jared Kaplan. Constitutional ai: Harmlessness from ai feedback, 2022. URL https://arxiv.org/abs/2212.08073.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 53728–53741. Curran Associates, Inc., 2023.

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. Kto: Model alignment as prospect theoretic optimization, 2024. URL https://arxiv.org/abs/2402.01306.

Jiwoo Hong, Noah Lee, and James Thorne. Orpo: Monolithic preference optimization without reference model, 2024. URL https://arxiv.org/abs/2403.07691.

Wenyi Xiao, Zechuan Wang, Leilei Gan, Shuai Zhao, Zongrui Li, Ruirui Lei, Wanggui He, Luu Anh Tuan, Long Chen, Hao Jiang, Zhou Zhao, and Fei Wu. A comprehensive survey of direct preference optimization: Datasets, theories, variants, and applications, 2025. URL https://arxiv.org/abs/2410.15595.

Ruili Jiang, Kehai Chen, Xuefeng Bai, Zhixuan He, Juntao Li, Muyun Yang, Tiejun Zhao, Liqiang Nie, and Min Zhang. A survey on human preference learning for aligning large language models. *ACM Comput. Surv.*, 58(6), December 2025. ISSN 0360-0300. doi: 10.1145/3773279. URL https://doi.org/10.1145/3773279.

Shicong Cen, Jincheng Mei, Katayoon Goshvadi, Hanjun Dai, Tong Yang, Sherry Yang, Dale Schuurmans, Yuejie Chi, and Bo Dai. Value-incentivized preference optimization: A unified approach to online and offline rlhf, 2025. URL https://arxiv.org/abs/2405.19320.

Alex Graves. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*, 2016.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers, 2019. URL https://arxiv.org/abs/1807.03819.

Shawn Tan, Yikang Shen, Zhenfang Chen, Aaron Courville, and Chuang Gan. Sparse universal transformer, 2023. URL https://arxiv.org/abs/2310.07096.

Andrea Banino, Jan Balaguer, and Charles Blundell. Pondernet: Learning to ponder, 2021. URL https://arxiv.org/abs/2107.05407.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. Transformer-XL: Attentive language models beyond a fixed-length context. In Anna Korhonen, David Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1285. URL https://aclanthology.org/P19-1285/.

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, and Timothy P. Lillicrap. Compressive transformers for long-range sequence modelling, 2019. URL https://arxiv.org/abs/1911.05507.

Aydar Bulatov, Yury Kuratov, and Mikhail Burtsev. Recurrent memory transformer. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 11079–11091. Curran Associates, Inc., 2022.

Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. Deebert: Dynamic early exiting for accelerating bert inference, 2020. URL https://arxiv.org/abs/2004.12993.

Wangchunshu Zhou, Canwen Xu, Tao Ge, Julian McAuley, Ke Xu, and Furu Wei. Bert loses patience: Fast and robust inference with early exit. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 18330–18341. Curran Associates, Inc., 2020.

Canwen Xu and Julian McAuley. A survey on dynamic neural networks for natural language processing. In Andreas Vlachos and Isabelle Augenstein, editors, *Findings of the Association for Computational Linguistics: EACL 2023*, pages 2370–2381, Dubrovnik, Croatia, May 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-eacl.180. URL https://aclanthology.org/2023.findings-eacl.180/.

Divya Jyoti Bajpai and Manjesh Kumar Hanawal. A survey of early exit deep neural networks in nlp, 2025. URL https://arxiv.org/abs/2501.07670.

David Raposo, Sam Ritter, Blake Richards, Timothy Lillicrap, Peter Conway Humphreys, and Adam Santoro. Mixture-of-depths: Dynamically allocating compute in transformer-based language models, 2024. URL https://arxiv.org/abs/2404.02258.

Yilong Chen, Junyuan Shang, Zhenyu Zhang, Yanxi Xie, Jiawei Sheng, Tingwen Liu, Shuohuan Wang, Yu Sun, Hua Wu, and Haifeng Wang. Inner thinking transformer: Leveraging dynamic depth scaling to foster adaptive internal thinking, 2025. URL https://arxiv.org/abs/2502.13842.

Bohong Wu, Mengzhao Chen, Xiang Luo, Shen Yan, Qifan Yu, Fan Xia, Tianqi Zhang, Hongrui Zhan, Zheng Zhong, Xun Zhou, Siyuan Qiao, and Xingyan Bin. Parallel loop transformer for efficient test-time computation scaling, 2025. URL https://arxiv.org/abs/2510.24824.

Ziyue Li, Yang Li, and Tianyi Zhou. Skip a layer or loop it? test-time depth adaptation of pretrained llms, 2025. URL https://arxiv.org/abs/2507.07996.

Sean McLeish, Ang Li, John Kirchenbauer, Dayal Singh Kalra, Brian R. Bartoldson, Bhavya Kailkhura, Avi Schwarzschild, Jonas Geiping, Tom Goldstein, and Micah Goldblum. Teaching pretrained language models to think deeper with retrofitted recurrence, 2025. URL https://arxiv.org/abs/2511.07384.

Boyi Zeng, He Li, Shixiang Song, Yixuan Wang, Ziwei He, Xinbing Wang, and Zhouhan Lin. Ponderlm-2: Pretraining llm with latent thoughts in continuous space, 2025. URL https://arxiv.org/abs/2509.23184.

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.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset, 2021b. URL https://arxiv.org/abs/2103.03874.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.emnlp-demos.6.

Sylvain Gugger, Lysandre Debut, Thomas Wolf, Philipp Schmid, Zachary Mueller, Sourab Mangrulkar, Marc Sun, and Benjamin Bossan. Accelerate: Training and inference at scale made simple, efficient and adaptable. https://github.com/huggingface/accelerate, 2022.

Qwen Team et al. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2(3), 2024.

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

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

A Limitations

While LoopRPT is effective for reinforcement pre-training of looped language models, this study has a few limitations that pave the way for future research: (1) Model Scale and Data Diversity: Due to computational constraints, our current instantiations are evaluated at the 1.4B and 2.6B parameter scales, with training primarily focused on the mathematical reasoning domain (OMNI-MATH). While our method shows strong transferability to coding and general logic, exploring the scaling laws of LoopRPT on much larger models (e.g., 7B+ parameters) and across broader, open-domain pre-training mixtures remains an important next step. (2) Fixed Maximum Loop Depth During Training: The current LoopRPT framework computes step-wise rewards and policy advantages over a predefined maximum computational horizon (e.g., $K = 4$ loops during training). Although the model learns an adaptive early-exit policy for inference, the fixed unrolling depth during training may cap the upper bound of reasoning complexity it can explore. Formulating a reinforcement objective that supports infinite-horizon or fully dynamic training depth is an exciting avenue for future work.

B Additional Theory

Setup and notation. For a fixed next-token position t in a sequence, LoopRPT defines: (i) a teacher (EMA) model $\bar{\theta}$ and a student model θ; (ii) a teacher exit distribution $\pi_{\bar{\theta}}(k)$ and student exit distribution $\pi_\theta(k)$ over latent steps $k \in \{1, \dots, K\}$ (Eq. (2–3)); (iii) a teacher reference step $t_{\text{ref}} = \min\{k : \sum_{j \le k} \pi_{\bar{\theta}}(j) \ge \tau\}$; (iv) a per-token dynamic baseline $b_{\text{ref}} = \ell_{\bar{\theta}}^{(t_{\text{ref}})}$ (Eq. (5)); (v) a step-wise reward $R(k) = \Delta_{\text{acc}}(k) - C(k)$ (Eq. (9)) where $\Delta_{\text{acc}}(k) = \ell_\theta^{(k)} - b_{\text{ref}}$ (Eq. (6)) and $C(k) = \lambda_t(k - t_{\text{ref}})$ (Eq. (7)), with λ_t derived from teacher uncertainty (Eq. (8)). We use $\Pi_\theta(k) \coloneqq \sum_{j \le k} \pi_\theta(j)$ to denote the CDF.

B.1 Why High-Entropy Token Selection Focuses Training Signal

Teacher entropy and selection. LoopRPT computes teacher entropy (Eq. (4)):

$$H_t \coloneqq -\sum_{v \in V} \bar{p}_{\bar{\theta}}(v \mid x_{<t}) \log \bar{p}_{\bar{\theta}}(v \mid x_{<t}), \tag{17}$$

and selects the top-ρ fraction of positions within each example as hard tokens (Sec. 3.2).

A principled proxy: expected gradient energy. We justify entropy selection by relating teacher uncertainty to the *expected squared norm* of the cross-entropy gradient, a standard proxy for how informative a sample is for learning.

Assumption B.1 (Teacher distribution as a proxy for label uncertainty)**.** Conditioned on context $x_{<t}$, the ground-truth next token y is drawn from the teacher distribution $\bar{p}(v) \coloneqq \bar{p}_{\bar{\theta}}(v \mid x_{<t})$.

Cross-entropy gradient. Let the student predictive distribution be $q(v) \coloneqq p_\theta(v \mid x_{<t})$. For a single sampled label y, the negative log-likelihood is $\mathcal{L}(q; y) = -\log q(y)$. Differentiating w.r.t. the student logits z (where $q = \text{Softmax}(z)$) yields the well-known form:

$$\nabla_z \mathcal{L}(q; y) = q - e_y, \tag{18}$$

where e_y is the one-hot vector.

Lemma B.2 (Expected gradient energy depends on collision probability)**.** *Under Assumption B.1, the expected squared ℓ_2-norm of the logit-gradient is*

$$\mathbb{E}_{y \sim \bar{p}}\big[\|\nabla_z \mathcal{L}(q; y)\|_2^2\big] = \|q\|_2^2 + 1 - 2\langle q, \bar{p}\rangle. \tag{19}$$

In particular, when the student matches the teacher locally ($q = \bar{p}$),

$$\mathbb{E}_{y \sim \bar{p}}\big[\|\nabla_z \mathcal{L}(\bar{p}; y)\|_2^2\big] = 1 - \|\bar{p}\|_2^2. \tag{20}$$

Proof. Using (18),

$$\|\nabla_z \mathcal{L}(q; y)\|_2^2 = \|q - e_y\|_2^2 = \|q\|_2^2 + \|e_y\|_2^2 - 2\langle q, e_y\rangle = \|q\|_2^2 + 1 - 2q(y).$$

Taking expectation over $y \sim \bar{p}$ gives $\mathbb{E}[q(y)] = \sum_v \bar{p}(v) q(v) = \langle q, \bar{p}\rangle$, which yields (19). Setting $q = \bar{p}$ gives (20). □

Relating collision probability to entropy. Define the collision probability $c(\bar{p}) \coloneqq \|\bar{p}\|_2^2 = \sum_v \bar{p}(v)^2$. The (order-2) Rényi entropy is $H_2(\bar{p}) \coloneqq -\log c(\bar{p})$, and it is standard that Shannon entropy upper-bounds Rényi-2 entropy:

$$H_t \geq H_2(\bar{p}) = -\log \|\bar{p}\|_2^2. \tag{21}$$

Equivalently,

$$\|\bar{p}\|_2^2 \leq e^{-H_t}. \tag{22}$$

Proposition B.3 (High entropy implies large expected gradient energy)**.** *Under Assumption B.1 and local student-teacher agreement ($q = \bar{p}$), the expected gradient energy satisfies*

$$\mathbb{E}_{y\sim\bar{p}}\big[\|\nabla_z \mathcal{L}(\bar{p}; y)\|_2^2\big] = 1 - \|\bar{p}\|_2^2 \;\geq\; 1 - e^{-H_t}. \tag{23}$$

Thus, larger teacher entropy H_t yields a (monotone) larger lower bound on gradient energy.

Proof. Combine (20) with (22): $1 - \|\bar{p}\|_2^2 \geq 1 - e^{-H_t}$. Monotonicity follows since $1 - e^{-H_t}$ increases in H_t. □

Connection to LoopRPT. LoopRPT selects top-ρ high-entropy tokens using (17). Proposition B.3 shows that, under a standard uncertainty-as-label-proxy assumption, higher teacher entropy provably implies larger expected cross-entropy gradient energy (and hence more informative updates) near local student-teacher agreement. This provides a principled explanation for why focusing losses (Sec. 3.3–3.4) on high-entropy positions improves sample-efficiency and amplifies useful training signals.

B.2 Noisy Latent Rollouts Optimize a Smoothed On-Policy Objective

Noisy latent rollouts. LoopRPT injects Gaussian noise into the recurrent latent states (Eq. (11)):

$$h^{(k)} \leftarrow h^{(k)} + \epsilon^{(k)}, \qquad \epsilon^{(k)} \sim \mathcal{N}(0, \sigma^2 I). \tag{24}$$

Let $\epsilon \coloneqq \{\epsilon^{(k)}\}_{k=1}^K$ denote the full noise trajectory, and write the resulting (noisy) exit distribution as $\pi_{\theta,\epsilon}(k)$. Under *reward regain*, the per-step student log-probability is recomputed on the same noisy trajectory, denoted $\ell_{\theta,\epsilon}^{(k)}$, and the step-wise reward is

$$R_\epsilon(k) \coloneqq \underbrace{\big(\ell_{\theta,\epsilon}^{(k)} - b_{\mathrm{ref}}\big)}_{\Delta_{\mathrm{acc},\epsilon}(k)} - \underbrace{\lambda_t(k - t_{\mathrm{ref}})}_{C(k)}. \tag{25}$$

This mirrors Eq. (6–9) with $\ell_\theta^{(k)}$ replaced by $\ell_{\theta,\epsilon}^{(k)}$.

Definition B.4 (Smoothed on-policy objective induced by latent noise)**.** Define

$$J_\sigma(\theta) \coloneqq \mathbb{E}_{\epsilon\sim\mathcal{N}(0,\sigma^2 I)}\left[\sum_{k=1}^K \pi_{\theta,\epsilon}(k)\, R_\epsilon(k)\right] = \mathbb{E}_\epsilon\, \mathbb{E}_{k\sim\pi_{\theta,\epsilon}}\big[R_\epsilon(k)\big]. \tag{26}$$

Key technical point. In the policy-gradient term of LoopRPT (Eq. (13)), $R_\epsilon(k)$ is used as a scalar reward signal. Accordingly, we analyze the score-function gradient, i.e., we do not backpropagate through the internal computation of $R_\epsilon(k)$.

Proposition B.5 (Exact score-function gradient of J_σ)**.** *Under the above convention (treating $R_\epsilon(k)$ as a reward signal), the gradient of J_σ is*

$$\nabla_\theta J_\sigma(\theta) = \mathbb{E}_\epsilon\, \mathbb{E}_{k\sim\pi_{\theta,\epsilon}}\Big[R_\epsilon(k)\, \nabla_\theta \log \pi_{\theta,\epsilon}(k)\Big]. \tag{27}$$

Consequently, Monte Carlo estimation using G i.i.d. noisy rollouts yields an unbiased estimator of the right-hand side of (27)*.*

Proof. By Definition B.4,

$$J_\sigma(\theta) = \mathbb{E}_\epsilon \sum_{k=1}^K \pi_{\theta,\epsilon}(k) R_\epsilon(k).$$

Treat $R_\epsilon(k)$ as constant w.r.t. θ in this gradient (score-function convention):

$$\nabla_\theta J_\sigma(\theta) = \mathbb{E}_\epsilon \sum_{k=1}^{K} R_\epsilon(k)\, \nabla_\theta \pi_{\theta,\epsilon}(k).$$

Using $\nabla_\theta \pi = \pi \nabla_\theta \log \pi$ gives

$$\nabla_\theta J_\sigma(\theta) = \mathbb{E}_\epsilon \sum_{k=1}^{K} \pi_{\theta,\epsilon}(k) R_\epsilon(k)\, \nabla_\theta \log \pi_{\theta,\epsilon}(k) = \mathbb{E}_\epsilon\, \mathbb{E}_{k\sim\pi_{\theta,\epsilon}} \left[R_\epsilon(k) \nabla_\theta \log \pi_{\theta,\epsilon}(k) \right],$$

which is (27). Unbiasedness of the Monte Carlo estimate follows from linearity of expectation. □

Proposition B.6 (Gaussian smoothing yields robustness bounds)**.** *Assume for a fixed θ that the per-noise rollout objective $F(\epsilon) \coloneqq \mathbb{E}_{k\sim\pi_{\theta,\epsilon}}[R_\epsilon(k)]$ is L-Lipschitz in ϵ: $|F(\epsilon) - F(\epsilon')| \le L\|\epsilon - \epsilon'\|$. Then*

$$\left|J_\sigma(\theta) - F(0)\right| = \left|\mathbb{E}_\epsilon[F(\epsilon)] - F(0)\right| \le L\, \mathbb{E}\|\epsilon\|. \tag{28}$$

Moreover, for $\epsilon \sim \mathcal{N}(0, \sigma^2 I_d)$, we have the explicit bound $\mathbb{E}\|\epsilon\| \le \sigma\sqrt{d}$, hence $|J_\sigma(\theta) - F(0)| \le L\sigma\sqrt{d}$.

Proof. By Jensen and Lipschitzness, $|\mathbb{E}F(\epsilon) - F(0)| \le \mathbb{E}|F(\epsilon) - F(0)| \le L\mathbb{E}\|\epsilon\|$, which is (28). For a Gaussian vector in $\mathbb{R}^d$, $\mathbb{E}\|\epsilon\| \le \sqrt{\mathbb{E}\|\epsilon\|^2} = \sqrt{\mathbb{E}\sum_{i=1}^d \epsilon_i^2} = \sigma\sqrt{d}$. □

Connection to LoopRPT. LoopRPT's noisy latent rollouts (Eq. (13)) induce the smoothed on-policy objective J_σ in (26). Proposition B.5 formalizes that, under the standard score-function convention used in the policy-gradient term, the rollout gradient estimates the exact gradient of this smoothed objective. Proposition B.6 further shows that latent Gaussian noise replaces a potentially brittle deterministic rollout objective $F(0)$ by its Gaussian average, which is robust to local perturbations with a deviation controlled by σ; this provides a theoretical justification for improved stability when learning exit behaviors from latent-space variability.

B.3 EMA Teacher as a Provably Slowly Moving Reference

EMA update. LoopRPT maintains an EMA teacher updated after each student step:

$$\bar{\theta}_n = \phi\bar{\theta}_{n-1} + (1-\phi)\theta_n, \qquad \phi \in [0, 1). \tag{29}$$

Lemma B.7 (Closed form and lag decomposition)**.** *For any $n \ge 1$,*

$$\bar{\theta}_n = \phi^n \bar{\theta}_0 + (1-\phi) \sum_{i=1}^{n} \phi^{n-i} \theta_i. \tag{30}$$

Moreover, defining parameter increments $\Delta_i \coloneqq \theta_i - \theta_{i-1}$, we have the exact identity

$$\theta_n - \bar{\theta}_n = \phi^n(\theta_0 - \bar{\theta}_0) + \sum_{i=1}^{n} \phi^{n-i} \Delta_i. \tag{31}$$

Proof. Unrolling (29) yields (30) by induction. For (31), subtract (30) from θ_n:

$$\theta_n - \bar{\theta}_n = \theta_n - \phi^n \bar{\theta}_0 - (1-\phi) \sum_{i=1}^{n} \phi^{n-i} \theta_i.$$

Rewrite θ_n as $\phi^n\theta_0 + \sum_{i=1}^n \phi^{n-i}(\theta_i - \phi\theta_{i-1})$ and collect terms to obtain the stated decomposition in Δ_i (details follow from telescoping). □

Interpretation. Eq. (31) shows the teacher *lags* behind the student by a geometrically weighted sum of recent student updates $\{\Delta_i\}$; hence $\bar{\theta}$ is a low-pass filtered version of θ.

Assumption B.8 (Lipschitz mapping from parameters to exit CDF)**.** For any $k \in \{1, \dots, K\}$ and any context $x_{<t}$, there exists $L_\Pi > 0$ such that

$$\left|\Pi_\theta(k) - \Pi_{\theta'}(k)\right| \le L_\Pi \|\theta - \theta'\|. \tag{32}$$

Proposition B.9 (Teacher reference step is stable under small drift)**.** *Fix $(x_{<t})$ and suppose for some $k^\star$ there is a margin $m > 0$ such that*

$$\Pi_{\bar{\theta}_{n-1}}(k^\star - 1) \le \tau - m \quad \textit{and} \quad \Pi_{\bar{\theta}_{n-1}}(k^\star) \ge \tau + m. \tag{33}$$

If $\|\bar{\theta}_n - \bar{\theta}_{n-1}\| \le \frac{m}{L_\Pi}$, then the teacher reference step computed from $\bar{\theta}_n$ remains unchanged: $t_{\mathrm{ref}}(\bar{\theta}_n) = k^\star$.

Proof. By Assumption B.8, for any k, $|\Pi_{\bar{\theta}_n}(k) - \Pi_{\bar{\theta}_{n-1}}(k)| \le L_\Pi \|\bar{\theta}_n - \bar{\theta}_{n-1}\| \le m$. Hence

$$\Pi_{\bar{\theta}_n}(k^\star - 1) \le \Pi_{\bar{\theta}_{n-1}}(k^\star - 1) + m \le (\tau - m) + m = \tau,$$

and

$$\Pi_{\bar{\theta}_n}(k^\star) \ge \Pi_{\bar{\theta}_{n-1}}(k^\star) - m \ge (\tau + m) - m = \tau.$$

Therefore the minimum k such that $\Pi_{\bar{\theta}_n}(k) \ge \tau$ is still $k^\star$. □

Proposition B.10 (EMA yields $O(1-\phi)$ consecutive-teacher drift)**.** *For the EMA update* (29)*,*

$$\|\bar{\theta}_n - \bar{\theta}_{n-1}\| = (1-\phi)\|\theta_n - \bar{\theta}_{n-1}\|. \tag{34}$$

In particular, when θ_n and $\bar{\theta}_{n-1}$ are close (typical in late training), the teacher drift is suppressed by the factor $(1-\phi)$.

Proof. Subtract $\bar{\theta}_{n-1}$ from (29): $\bar{\theta}_n - \bar{\theta}_{n-1} = (1-\phi)(\theta_n - \bar{\theta}_{n-1})$. Taking norms yields (34). □

Connection to LoopRPT. LoopRPT computes t_{ref} (Sec. 3.3) and $b_{\mathrm{ref}} = \ell_{\bar{\theta}}^{(t_{\mathrm{ref}})}$ (Eq. (5)) from the teacher. Proposition B.10 shows that EMA makes the teacher a slowly moving reference, and Proposition B.9 further implies that whenever the teacher CDF crosses the exit threshold with a nontrivial margin, the discrete reference step t_{ref} is invariant to small parameter drift. Together, these results formalize why EMA reduces "target chasing" when jointly optimizing the exit policy and intermediate representations.

C Online Hard-token Selection Details

We perform entropy-based filtering on-the-fly and apply reinforcement-style losses only to the top-ρ high-entropy tokens, following the high-entropy minority token selection strategy in Wang et al. [2025]. Concretely, we compute token entropies using the teacher's *final* latent step, which reflects the model's most refined uncertainty estimate for next-token prediction.

Let $\mathbf{h}_{\bar{\theta}}^{(K)}$ be the teacher hidden state at the last loop step. We compute logits and shift them for next-token alignment:

$$\mathbf{z}_{\bar{\theta}}^{(K)} = \mathrm{LMHead}(\mathbf{h}_{\bar{\theta}}^{(K)}) \in \mathbb{R}^{B\times S\times|\mathcal{V}|}, \qquad \tilde{\mathbf{z}} = \mathbf{z}_{\bar{\theta}}^{(K)}(:,\, 1:-1,\, :) \in \mathbb{R}^{B\times(S-1)\times|\mathcal{V}|}. \tag{35}$$

The per-token entropy is then

$$H_{i,t} = -\sum_{v\in\mathcal{V}} p_{i,t}(v)\log p_{i,t}(v), \quad p_{i,t} = \mathrm{Softmax}(\tilde{\mathbf{z}}_{i,t,:}). \tag{36}$$

We restrict selection to valid training positions using the shifted loss mask $\mathbf{m} \in \{0,1\}^{B\times(S-1)}$ (response tokens). For each example i, we compute a *row-wise* quantile threshold over valid positions,

$$q_i = \mathrm{Quantile}\Big(\{H_{i,t} : \mathbf{m}_{i,t} = 1\},\, 1-\rho\Big), \tag{37}$$

and define the entropy mask

$$\mathbf{m}_{\mathrm{RPT},i,t} = \mathbb{I}\big[H_{i,t} > q_i\big]. \tag{38}$$

Finally, we update the effective training mask by elementwise multiplication,

$$\mathbf{m} \leftarrow \mathbf{m} \odot \mathbf{m}_{\mathrm{RPT}}, \tag{39}$$

so all subsequent losses are computed only on the selected high-entropy positions.

D K3-style KL surrogate

We regularize the student against the EMA teacher using the same stable surrogate. For a given step k, let $\ell_\theta^{(k)}$ and $\ell_{\bar{\theta}}^{(k)}$ be the student/teacher log-probabilities of the gold next token. Define the log-ratio

$$\Delta^{(k)} = \ell_{\bar{\theta}}^{(k)} - \ell_\theta^{(k)}. \tag{40}$$

Optionally, we clamp $\Delta^{(k)}$ to a bounded range. We then compute

$$u^{(k)} = \exp(\Delta^{(k)}), \qquad \mathrm{K3}^{(k)} = u^{(k)} - \Delta^{(k)} - 1, \tag{41}$$

and clamp $\mathrm{K3}^{(k)}$ to stabilize optimization. Finally, we weight the per-step KL surrogate on the noisy rollout forward pass using the rollout masks $\mathbf{m}^G$, and average over masked tokens:

$$\mathcal{L}_{KL} = \frac{\sum\left(\sum_k \mathrm{K3}^{(k)}\right) \odot \mathbf{m}^G}{\sum \mathbf{m}^G + \epsilon}. \tag{42}$$

E Additional Experimental Details

E.1 Details of Datasets

MMLU [Hendrycks et al., 2021a] MMLU[10] is a multi-domain multiple-choice benchmark covering 57 subjects, designed to test broad factual knowledge and general reasoning.

MMLU-Pro [Wang et al., 2024] MMLU-Pro[11] is an updated variant of MMLU with more challenging questions and stricter evaluation protocols to reduce ambiguity and contamination effects.

BBH [Suzgun et al., 2023] BIG-Bench Hard (BBH)[12] is a curated subset of BIG-Bench tasks emphasizing compositional and multi-step reasoning (e.g., logical deduction, symbolic manipulation).

ARC-Challenge (ARC-C) [Clark et al., 2018] ARC-C[13] is a grade-school science multiple-choice benchmark focusing on difficult questions that require reasoning beyond surface pattern matching.

HellaSwag [Zellers et al., 2019] HellaSwag[14] evaluates commonsense narrative completion by selecting the most plausible continuation among candidates.

Winogrande [Sakaguchi et al., 2021] Winogrande[15] is a large-scale pronoun resolution benchmark that tests commonsense reasoning and bias-sensitive coreference decisions.

GSM8K [Cobbe et al., 2021] GSM8K[16] is a grade-school math word-problem benchmark that requires multi-step arithmetic reasoning.

MATH [Hendrycks et al., 2021b] MATH [17] is a competition-mathematics dataset with problems labeled by difficulty Level 1–5. We evaluate on the full 5,000-problem test split and on **MATH500**[18], a 500-problem subset drawn from the MATH competition dataset. Both use the official problem-level labels for the difficulty breakdown in Fig. 4.

AIME The American Invitational Mathematics Examination[19] provides 30 short-answer olympiad problems per year, which we use to probe behavior furthest from the training distribution. We report AIME 2024 and AIME 2025.

[10] https://huggingface.co/datasets/cais/mmlu
[11] https://huggingface.co/datasets/TIGER-Lab/MMLU-Pro
[12] https://github.com/suzgunmirac/BIG-Bench-Hard
[13] https://huggingface.co/datasets/allenai/ai2_arc
[14] https://huggingface.co/datasets/Rowan/hellaswag
[15] https://huggingface.co/datasets/allenai/winogrande
[16] https://huggingface.co/datasets/openai/gsm8k
[17] https://huggingface.co/datasets/hendrycks/math
[18] https://huggingface.co/datasets/HuggingFaceH4/MATH-500
[19] https://huggingface.co/datasets/AI-MO/aimo-validation-aime

Table 6: Hyperparameters used for LoopRPT training.

Params	Values
Gradient clip norm	1.0
Teacher update momentum ϕ	0.995
Batch size	8
Epoch number	3
Rollout number	8
Learning rate	5×10^{-6}
AdamW	$(0.9, 0.999)$
Weight decay	0.01
Warmup ratio	0.03
Max sequence length	4096
Top-Entropy ratio ρ	0.2
Time penalty base coefficient λ_{base}	0.02
Time penalty scale coefficient λ_{scale}	0.5
Policy-gradient loss coefficient α	1.0
Backbone loss coefficient β	1.0
KL loss coefficient δ	0.0001
Entropy loss coefficient γ	0.01

Table 7: Evaluation settings and frameworks of benchmarks. Greedy decoding uses no chat template and the maximum-loop or adaptive-early-exit setting.

Benchmark	Settings	Framework
General		
MMLU	logprobs, 5-shot	lm-eval-harness
MMLU-Pro	strict match, 5-shot CoT	lm-eval-harness
BBH	strict match, 3-shot CoT	lm-eval-harness
MMLU (no-CoT)	logprobs, 0-shot	lm-eval-harness
BBH (no-CoT)	strict match, 0-shot	lm-eval-harness
ARC-C	logprobs, 25-shot	lm-eval-harness
HellaSwag	logprobs, 10-shot	lm-eval-harness
Winogrande	logprobs, 5-shot	lm-eval-harness
Math		
GSM8k	strict match, 3-shot CoT	lm-eval-harness
GSM8k (no-CoT)	strict match, 0-shot	lm-eval-harness
MATH500	math_verify, greedy	lm-eval-harness
MATH (full)	math_verify, greedy	lm-eval-harness
AIME 2024/2025	exact match, greedy	lm-eval-harness
Code		
HumanEval	pass@1	evalplus
HumanEval+	pass@1	evalplus
MBPP	pass@1	evalplus
MBPP+	pass@1	evalplus

OpenThoughts-3 **OpenThoughts-3**[20] is a multi-domain reasoning corpus of roughly 100k examples spanning mathematics, code, and science at a 5:3:2 ratio. We use it as the broader training corpus in Appendix F.3.

HumanEval [Chen et al., 2021] HumanEval[21] is a code-generation benchmark consisting of programming problems with unit-test based evaluation.

HumanEval+ [Liu et al., 2023] HumanEval+[22] extends HumanEval with additional and harder tests to reduce overfitting to the original unit tests and better measure functional correctness.

MBPP [Austin et al., 2021] MBPP (Mostly Basic Programming Problems)[23] is a Python programming benchmark covering short, diverse coding tasks with reference tests.

MBPP+ [Liu et al., 2023] MBPP+[24] strengthens MBPP by adding extended test suites, improving robustness of functional evaluation.

Omni-Math [Gao et al., 2024a] OMNI-MATH [25] is a benchmark for evaluating mathematical reasoning in large language models, composed of problems drawn from international mathematics competitions. The dataset emphasizes multi-step reasoning and symbolic manipulation.

E.2 Training Hyperparameters and Infrastructure

We implement our training pipeline using Hugging Face [Wolf et al., 2020] and Accelerate [Gugger et al., 2022], and speed up training with Distributed Data Parallel, mixed precision, and gradient checkpointing. We train for 3 epochs with a maximum sequence length of 4096 and use the AdamW optimizer. We apply a cosine decay learning rate schedule with a warmup phase to stabilize training. For each training example, we independently sample 8 times from an isotropic Gaussian distribution, $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$ with $\sigma = 0.1$ (i.e., variance 0.01), yielding 8 distinct OURO latent trajectories. All

[20] https://huggingface.co/datasets/open-thoughts/OpenThoughts3-1.2M
[21] https://huggingface.co/datasets/openai/openai_humaneval
[22] https://huggingface.co/datasets/evalplus/humanevalplus
[23] https://huggingface.co/datasets/google-research-datasets/mbpp
[24] https://huggingface.co/datasets/evalplus/mbppplus
[25] https://huggingface.co/datasets/KbsdJames/Omni-MATH

Table 8: Compute-matched controls for Ouro-1.4B. NTP is continued next-token prediction (SFT). Best per row in **bold**.

Benchmark	Base	+NTP	+GRPO	+LoopRPT
NTR-Easy	74.40	74.59	73.62	**75.00**
NTR-Med	57.20	56.63	56.44	**57.72**
NTR-Hard	33.91	34.29	33.00	**34.82**
MMLU	67.35	**67.82**	67.60	67.62
MMLU-Pro	48.64	48.89	48.57	**49.21**
ARC-C	60.95	60.58	49.23	**61.10**
HellaSwag	74.30	74.51	74.36	**74.93**
WinoGrande	71.67	71.19	66.93	**71.98**
BBH	70.97	71.10	28.84	**71.37**
GSM8K	78.17	77.63	4.17	**78.92**
MATH500	20.60	29.40	12.60	**37.00**
HumanEval	69.47	67.07	46.07	**69.51**
HumanEval+	67.01	64.63	42.86	**67.07**
MBPP	73.02	73.02	31.36	**75.40**
MBPP+	60.85	62.17	20.58	**63.76**

Table 9: Compute-matched controls for Ouro-2.6B. NTP is continued next-token prediction (SFT). Best per row in **bold**.

Benchmark	Base	+NTP	+GRPO	+LoopRPT
NTR-Easy	74.51	75.80	74.20	**76.07**
NTR-Med	57.35	58.83	56.58	**60.21**
NTR-Hard	34.35	35.21	33.00	**37.24**
MMLU	73.85	**74.00**	73.60	73.91
MMLU-Pro	54.07	53.96	53.84	**54.19**
ARC-C	66.12	66.75	49.89	**66.89**
HellaSwag	79.28	79.74	78.07	**80.03**
WinoGrande	75.85	**76.58**	67.19	76.47
BBH	77.98	78.19	29.36	**78.24**
GSM8K	81.76	82.41	6.63	**85.36**
MATH500	43.00	43.20	18.00	**43.50**
HumanEval	75.02	75.61	49.88	**76.83**
HumanEval+	70.13	68.29	46.17	**71.95**
MBPP	77.10	76.72	61.10	**77.24**
MBPP+	64.81	63.76	57.58	**65.08**

hyperparameters are summarized in Table 6. All experiments are conducted on 8 NVIDIA A100 GPUs with 80GB memory. Each LoopRPT run takes approximately 2 hours for Ouro-1.4B and around 4 hours for Ouro-2.6B.

E.3 Detailed Evaluation Settings

We evaluated the LoopRPT-trained Ouro model across general knowledge, reasoning, mathematics, science, coding, and multilingual capabilities, focusing on its improvements over the original model. We also compared it with open-source base models of similar scale to Ouro, including the Qwen2.5 [Team et al., 2024], Qwen3 [Yang et al., 2025], Gemma3 [Team et al., 2025], LLaMA3.1 [Grattafiori et al., 2024], and LLaMA3.2 [Grattafiori et al., 2024] series. Following Ouro [Zhu et al., 2025], all evaluations were conducted using lm-eval-harness [Gao et al., 2024b] and evalplus [Liu et al., 2023]. Detailed evaluation settings and metrics are provided in Table 7.

F Additional Discussion

F.1 Compute-Matched Controls

Our main-text controls (Table 3) change only the training objective while holding the checkpoint, data, step budget, and schedule fixed. We use two objectives as controls. *Continued-NTP* applies the standard next-token-prediction loss to the same OMNI-MATH sequences, which estimates the contribution of additional training on the corpus. *GRPO/RLVR* applies group-relative policy optimization [Shao et al., 2024] on output-token rewards, which estimates the contribution of an off-the-shelf RL recipe. Table 8 and Table 9 report the full matrix, including the per-difficulty NTR splits, at both scales.

Two patterns emerge. First, Continued-NTP stays flat relative to the base checkpoint on the general benchmarks and on NTR, so a further pass over the corpus is not the source of the improvement. Second, GRPO fails to converge on the looped backbone. The multi-step benchmarks collapse (GSM8K, BBH, ARC-C, and MBPP+), the NTR splits stay flat, and the knowledge-heavy multiple-choice tasks (MMLU and MMLU-Pro) hold. This matches the argument in Sec. 3 that a sparse output-token reward provides no credit assignment over latent iterations. The remaining LoopRPT-over-Continued-NTP gap therefore measures the contribution of the step-wise latent objective.

F.2 No-CoT Zero-Shot Evaluation

GSM8K and BBH use few-shot chain-of-thought in the main evaluation. Table 10 repeats them with the scaffold removed (few-shot $= 0$, GSM8K answers directly, BBH uses the no-CoT "Q:/A:" template), alongside MMLU, which never uses chain-of-thought. Removing the scaffold lowers accuracy sharply on GSM8K and BBH, so the few-shot CoT protocol overstates a looped model's

Table 10: No-CoT zero-shot accuracy (few-shot = 0; GSM8K answers directly and BBH uses the no-CoT template). Subscripts give the change over Ouro.

Scale	Benchmark	Ouro	LoopRPT
1.4B	MMLU	68.32	68.32
	GSM8K	29.87	43.52$_{+13.65}$
	BBH	38.46	39.26$_{+0.80}$
2.6B	MMLU	73.76	73.06$_{-0.70}$
	GSM8K	41.32	67.32$_{+26.00}$
	BBH	54.17	56.84$_{+2.67}$

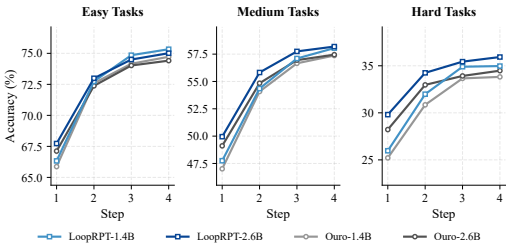


Figure 7: Per-step next-token reasoning accuracy across difficulty buckets.

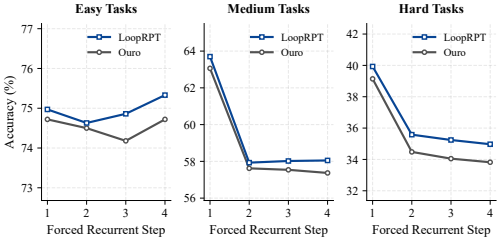


Figure 8: Forced-depth evaluation across difficulty buckets.

internal reasoning. The LoopRPT gain nonetheless survives: no-CoT GSM8K improves by 13.7 points at 1.4B and 26.0 at 2.6B, while MMLU barely moves.

F.3 Broader-Corpus Scaling

Table 11: Broader-corpus scaling on OpenThoughts-3 (100k, multi-domain) versus the narrow-domain OMNI-MATH run, Ouro-1.4B, same LoopRPT recipe. The downstream protocol follows Appendix E.

Model (training corpus)	MMLU	ARC-C	HellaSwag	WinoGrande	GSM8K	MBPP+
Ouro-1.4B base	67.35	60.95	74.30	71.67	78.17	60.85
+LoopRPT (OMNI-MATH)	67.62	61.10	74.93	71.98	78.92	63.76
+LoopRPT (OpenThoughts)	**67.83**	**61.19**	**75.66**	**72.24**	**79.03**	**63.88**

Our training corpus is a single narrow domain, so we also run the same LoopRPT recipe on **OpenThoughts-3** (100k examples, math:code:science $\approx$ 5:3:2), roughly 23$\times$ the OMNI-MATH corpus, on Ouro-1.4B. Table 11 reports the downstream result. All six benchmarks match or exceed the narrow-domain run, so the mechanism does not depend on the narrow math corpus. The calibrated early-exit behavior is also preserved: the Hard-bucket adaptive exit step is 3.12, close to the original 3.07 and below the base checkpoint.

F.4 Forced-depth Evaluation

We set the exit threshold to 1.0 and vary the latent steps to evaluate performance under compulsory computation. As shown in Fig. 8, forced-depth accuracy is not guaranteed to improve monotonically as depth increases, and may even degrade for Medium and Hard buckets. This behavior is expected for looped models: later latent steps are not explicitly optimized to monotonically refine token-level cross-entropy, and additional iterations can introduce distributional shifts or over-updating of latent representations when the model is forced to continue reasoning. Consequently, forced-depth evaluation measures the behavior under compulsory computation, rather than the intrinsic quality of intermediate representations.

LoopRPT consistently outperforms the baseline across all buckets and depths, indicating that reinforcement pre-training improves robustness even under forced computation.

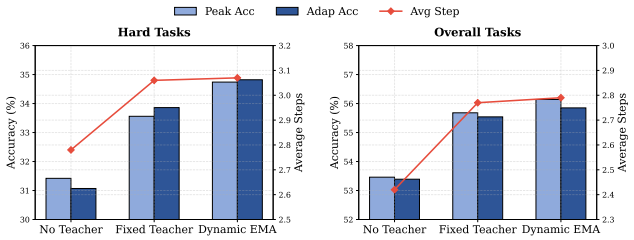


Figure 9: Performance comparison of different EMA teacher mechanisms on next-token reasoning tasks.

F.5 Analysis of the EMA Mechanism

To address the mechanism by which the EMA teacher stabilizes training and avoids potential self-distillation bottlenecks, we further ablate the teacher configuration. Figure 9 presents a comparison among three setups: standard LoopRPT (Dynamic EMA), Fixed Teacher ($\phi = 1.0$), and No Teacher (Remove the teacher baseline in $R(k)$ and eliminate the KL penalty).

When the reference teacher is removed, performance drops sharply across all metrics. Without a stable reference, the step-wise advantage $\widehat{A}(k)$ becomes erratic. The model attempts to maximize rewards by exploiting premature exits rather than refining latent reasoning, resulting in drastically reduced average steps coupled with a collapse in accuracy. The EMA teacher provides a consistent moving baseline that anchors the reward and prevents the exit policy from collapsing into low-entropy, shallow computations.

Using a Fixed Teacher prevents the extreme degradation seen in the No Teacher setup. However, it falls short of the Dynamic EMA configuration. A static teacher inherently bounds the exploration space, restricting the learning signal to patterns strictly within the initial checkpoint's capabilities. This imposes a "self-distillation upper limit." The Dynamic EMA, acting as a low-pass filter of the student's recent updates, continuously evolves. It pushes the reward baseline marginally higher as the student improves, incentivizing the model to discover reasoning paths that surpass the initial fixed reference while maintaining stable step-wise constraints.

F.6 Wall-clock Latency and Physical Efficiency

Figure 10 illustrates the average exit time (in milliseconds) required per token generation across varying context lengths on a single A100 GPU.

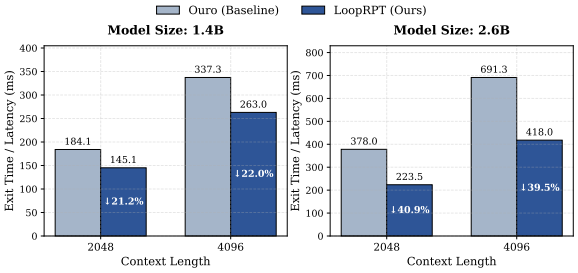


Figure 10: Comparison of actual physical inference time for token exiting.

Across both model scales and context lengths, LoopRPT consistently reduces physical latency. On Ouro-2.6B at a 4096-token context, LoopRPT cuts real-time latency by 39.5%. These results confirm that the early-exit calibration learned via our dense reinforcement pre-training is highly effective. The computational savings from terminating the latent recurrence early (fewer forward passes through the shared transformer block) vastly outweigh any systemic memory bandwidth overheads associated with the exit routing mechanism.

The relative latency reduction also holds as the model size scales from 1.4B to 2.6B, extending the efficiency gain to larger parameter regimes where latent computation costs are more pronounced.

F.7 Training Cost

LoopRPT adds three terms on top of Continued-NTP: an EMA teacher forward pass, a grouped noisy student forward pass, and per-step readouts of the shared backbone. The grouped noisy rollout (G=8)

dominates the overhead. The EMA teacher forward stores no activations, and the step-weighted readout reuses hidden states the backbone already produces. Measured per step on a single A100, LoopRPT costs $1.10\times$ GRPO at $G{=}8$, while a full Continued-NTP run takes 1.10 h (1.4B) and 2.14 h (2.6B) on $8\times$A100. The extra training cost is offset at inference, where the calibrated early exit lowers token latency by 22% (1.4B) and 39.5% (2.6B) at a 4096-token context (Fig. 10).

G Training Dynamics Visualization

Figure 11 and 12 visualizes the training dynamics of LoopRPT on Ouro. Subfigure (a) tracks validation next-token accuracy under maximum-loop inference (Peak) and adaptive early exit (Adap.), together with the average exit steps. Subfigure (b) shows the evolution of the student's exit behavior relative to the EMA teacher reference: we report rolling-mean reasoning steps and the student's within-window variability. Overall, the curves indicate that LoopRPT improves validation performance while progressively reducing the required reasoning steps, consistent with better early-exit calibration.

H Case Studies

To further investigate the underlying reasoning mechanisms of LoopRPT compared to the base Ouro model, we conduct a detailed qualitative analysis across three distinct domains: general reasoning, mathematical problem-solving, and code generation. As illustrated in Fig. 13, Fig. 14, and Fig. 15, these cases highlight the error-correction and multi-step dependency tracking capabilities of our model.

General reasoning & factual consistency. In complex scenarios requiring interdisciplinary knowledge (e.g., thermodynamics and biology), the base model frequently exhibits *factual hallucinations* or *logical fragility*. For instance, in Case 2 (Biology), the base model incorrectly categorizes birds as ectotherms despite mentioning their endothermic nature later in the trace. In contrast, LoopRPT maintains high factual consistency throughout the reasoning chain. In Case 3 (Discrete Logic), LoopRPT successfully identifies implicit logical equivalences between different Boolean expressions, whereas the base model fails to recognize the semantic overlap, leading to incomplete conclusions.

Mathematical logic & constraint satisfaction. The mathematical cases on GSM8K reveal that the base model is prone to *variable mapping errors* and *set-neglect*. In Case 1 (Scheduling), the base model confuses the count of inpatients with appointments, whereas LoopRPT meticulously tracks separate constraints. In Case 2 (Finance) and Case 3 (Fractional Calculation), the base model fails to maintain the global state of sets (e.g., neglecting the combined total of two agents). LoopRPT demonstrates a superior ability to iterate through multi-step arithmetic constraints, likely benefiting from the implicit loops that reinforce state tracking.

Algorithmic invariants in coding. On coding benchmarks (MBPP and HumanEval), the divergence is primarily seen in *boundary condition handling* and *algorithmic interpretation*. The base model often produces "off-by-one" errors (Case 2) or fails to identify cyclic invariants in array operations (Case 3). LoopRPT consistently generates code that adheres to task-specific constraints and rigorous logic. This suggests that the latent reasoning loops in LoopRPT act as a form of implicit verification, allowing the model to simulate execution states more effectively than standard autoregressive generation.

I Full Training Algorithm

Alg. 1–2 provides a description of our training iteration. The algorithm uses two phases: (i) a no-gradient phase that constructs a dense step-wise reward table for each selected token, and (ii) a gradient phase that jointly optimizes the exit policy via noisy latent rollouts and the backbone via step-weighted next-token learning.

Table 12: Comparison of 2.6B LoopLM with 3-12B parameter baselines. Results for vanilla LLMs are taken from Zhu et al. [2025], with Ouro results obtained via our reimplementation. Best in **bold**, second best underlined. Subscripts indicate LoopRPT's improvement over Ouro baseline.

Benchmark	Gemma3 3B	Llama3.2 3B	Qwen2.5 4B	Qwen3 4B	Qwen2.5 7B	Llama3.1 8B	Qwen3 8B	Gemma3 12B	Ouro 2.6B	**LoopRPT 2.6B**
General Tasks										
MMLU	65.62	59.69	73.19	58.37	74.20	73.02	**76.63**	72.14	73.85	73.91$_{+0.06}$
MMLU-Pro	37.87	33.34	51.40	34.61	43.55	43.24	53.72	49.21	54.07	**54.19**$_{+0.12}$
BBH	55.37	39.45	71.14	66.32	53.72	71.56	77.65	**78.41**	77.98	78.24$_{+0.26}$
ARC-C	55.46	52.47	63.65	60.75	63.65	60.75	66.10	**72.44**	66.12	66.89$_{+0.77}$
HellaSwag	74.54	73.09	75.66	75.58	79.98	81.97	79.60	**83.68**	79.28	80.03$_{+0.75}$
Winogrande	70.17	69.14	71.19	71.27	76.48	77.11	76.80	**77.74**	75.85	76.47$_{+0.62}$
Math & Coding Tasks										
GSM8K	74.60	67.20	72.86	68.69	81.50	78.17	83.09	77.18	81.76	**85.36**$_{+3.60}$
HumanEval	68.90	29.90	77.70	34.80	79.30	38.40	**84.80**	46.30	75.02	76.83$_{+1.81}$
HumanEval+	62.20	26.20	70.70	29.30	70.60	31.10	**75.30**	37.20	70.13	71.95$_{+1.82}$
MBPP	63.00	50.30	78.80	60.60	73.80	62.40	**79.00**	73.50	77.10	77.24$_{+0.14}$
MBPP+	54.20	39.70	65.90	51.10	63.50	51.60	**67.90**	66.10	64.81	65.08$_{+0.27}$

Algorithm 1 Phase I: Dense Step-wise Reward Table Construction with RPT Token Filtering

Require: Batch $\{x_{0:S}\}$ with valid-token mask $m \in \{0,1\}^{B\times(S-1)}$ (aligned to next-token positions); student θ; EMA teacher $\bar{\theta}$; max steps K; exit threshold τ; top-ratio ρ; $\lambda_{\text{base}}, \lambda_{\text{scale}}$; constant ε.
Ensure: RPT mask m_{RPT}; reward table $\{R_t(k)\}_{k=1}^{K}$; step advantage $\{A_{b,t}(k)\}_{k=1}^{K}$.

Teacher forward (for hard-token selection and reference quantities)
Run teacher LoopLM for K steps to obtain last-step logits $\bar{z}_t^{(K)}$ and gate logits $\{\bar{a}_t^{(k)}\}_{k=1}^{K}$ at each next-token position t.
Compute last-step entropy $H_t^{\text{last}} = -\sum_{v\in V} \bar{p}_t(v) \log \bar{p}_t(v)$ where $\bar{p}_t = \text{Softmax}(\bar{z}_t^{(K)})$.
RPT hard-token filtering (top-ρ entropy on valid positions)
for each example i in the batch **do**
    $q_i \leftarrow \text{Quantile}(\{H_{i,t}^{\text{last}} : m_{i,t} = 1\}, 1-\rho)$
    $m_{\text{RPT},i,t} \leftarrow \mathbb{I}[H_{i,t}^{\text{last}} > q_i]$
end for
$m \leftarrow m \odot m_{\text{RPT}}$
Reference step t_{ref} **and dynamic baseline** b_{ref}
Compute teacher exit distribution $\pi_{\bar{\theta}}(k)$ via Eq. (2–3) from $\{\bar{a}_t^{(k)}\}$ and its CDF $\Pi_{\bar{\theta}}(k) = \sum_{j\le k} \pi_{\bar{\theta}}(j)$.
$t_{\text{ref}} \leftarrow \min\{k : \Pi_{\bar{\theta}}(k) \ge \tau\}$ (use K if never exceeded).
Compute teacher per-step gold log-prob $\ell_{\bar{\theta}}^{(k)}(t) = \log p_{\bar{\theta}}(x_t \mid x_{<t}; \bar{h}_t^{(k)})$.
Set $b_{\text{ref}}(t) \leftarrow \ell_{\bar{\theta}}^{(t_{\text{ref}})}(t)$ ▷ Eq. 5
Difficulty-aware time penalty weight λ_t
Compute teacher entropy at the reference step H_t^{ref} (from teacher distribution at t_{ref}).
$d_t \leftarrow \text{Clamp}\big(H_t^{\text{ref}} / \log|V|,\ 0,\ 1\big), \quad \lambda_t \leftarrow \lambda_{\text{base}}\,(1 + \lambda_{\text{scale}}(1-d_t))$ ▷ Eq. 8
Student deterministic pass for step-wise rewards
Run student without noise to obtain $\ell_\theta^{(k)}(t) = \log p_\theta(x_t \mid x_{<t}; h_t^{(k)})$ for $k = 1, \ldots, K$.
for $k = 1, \ldots, K$ **do**
    $\Delta_{\text{acc},t}(k) \leftarrow \ell_\theta^{(k)}(t) - b_{\text{ref}}(t)$ ▷ Eq. 6
    $C_t(k) \leftarrow \lambda_t\,(k - t_{\text{ref}})$ ▷ Eq. 7
    $R_t(k) \leftarrow \Delta_{\text{acc},t}(k) - C_t(k)$ ▷ Eq. 9
end for
Step advantage for representation learning
$\mu_{R,t} \leftarrow \frac{1}{K}\sum_{j=1}^{K} R_t(j), \quad \sigma_{R,t} \leftarrow \sqrt{\frac{1}{K}\sum_{j=1}^{K}(R_t(j) - \mu_{R,t})^2}$
for $k = 1, \ldots, K$ **do**
    $A_{b,t}(k) \leftarrow \frac{R_t(k) - \mu_{R,t}}{\sigma_{R,t} + \varepsilon}$ ▷ Eq. 10
end for
return $(m_{\text{RPT}}, \{R_t(k)\}_{k=1}^{K}, \{A_{b,t}(k)\}_{k=1}^{K})$ on positions where $m = 1$.

Algorithm 2 Phase II: Noisy Rollout Policy Gradient + Step-Weighted Next-Token Training

Require: Batch $\{x_{0:S}\}$ with effective mask m (after Alg. 1); student θ, EMA teacher $\bar{\theta}$; max steps K, threshold τ; reference quantities from Phase I: t_{ref}, $b_{\text{ref}}(t)$, λ_t, and $A_{b,t}(k)$; rollouts G, noise scale σ; loss weights $\alpha, \beta, \gamma, \delta$; constant ε.

Ensure: Updated θ and EMA teacher $\bar{\theta}$.

Deterministic student pass (for π_θ and step-weighted NTP)

Run student without noise to obtain gate logits $\{a_t^{(k)}\}_{k=1}^K$ and per-step gold log-probs $\ell_\theta^{(k)}(t)$.

Compute exit distribution $\pi_\theta(k)$ via Eq. (2–3).

for $k = 1, \ldots, K$ **do**

  $w_t(k) \leftarrow \text{softmax}\Big[\pi_\theta(k)\Big(1 + \text{ReLU}(A_{b,t}(k))\Big)\Big]$ ▷ Eq. 15

end for

$L_{\text{rep}} \leftarrow -\sum_{k=1}^K w_t(k)\, \ell_\theta^{(k)}(t)$ ▷ Eq. 14

(Applied only on positions where $m = 1$; sums/means are over those positions.)

Entropy regularization on exit distribution

$L_{\text{ent}} \leftarrow -\mathbb{E}\Big[\sum_{k=1}^K \pi_\theta(k) \log \pi_\theta(k)\Big]$.

Noisy latent rollouts with reward regain (on-policy reward recomputation)

for $g = 1, \ldots, G$ **do**

  Run a noisy recurrence (Eq. 11) to obtain noisy states $\{h_{t,g}^{(k)}\}_{k=1}^K$ and rollout gates $\{a_{t,g}^{(k)}\}_{k=1}^K$.

  Compute rollout exit distribution $\pi_\theta^{(g)}(k)$ from $\{a_{t,g}^{(k)}\}$.

  Recompute per-step gold log-probs on rollout states: $\ell_{\theta,g}^{(k)}(t) = \log p_\theta(x_t \mid x_{<t}; h_{t,g}^{(k)})$.

  for $k = 1, \ldots, K$ **do**

    $\Delta_{\text{acc},t}^{(g)}(k) \leftarrow \ell_{\theta,g}^{(k)}(t) - b_{\text{ref}}(t)$

    $C_t(k) \leftarrow \lambda_t\,(k - t_{\text{ref}})$

    $R_t^{(g)}(k) \leftarrow \Delta_{\text{acc},t}^{(g)}(k) - C_t(k)$

  end for

  Sample exit step $t^{(g)} \sim \pi_\theta^{(g)}(\cdot)$ and set rollout reward $r^{(g)} \leftarrow R_t^{(g)}(t^{(g)})$.

end for

Group-normalized rollout advantage and policy gradient

$A^{(g)} \leftarrow \dfrac{r^{(g)} - \text{mean}_g[r^{(g)}]}{\text{std}_g[r^{(g)}] + \varepsilon}$ ▷ Eq. 12

$L_{\text{PG}} \leftarrow -\mathbb{E}_g\Big[A^{(g)}\, \log \pi_\theta^{(g)}\big(t^{(g)}\big)\Big]$ ▷ Eq. 13

Rollout KL regularization (K3 surrogate, rollout aggregation)

Compute teacher per-step gold log-probs $\ell_{\bar{\theta}}^{(k)}(t)$ on the same rollout-evaluated tokens/steps.

for $k = 1, \ldots, K$ **do**

  $\Delta^{(k)} \leftarrow \ell_{\bar{\theta}}^{(k)}(t) - \ell_{\theta,g}^{(k)}(t)$ ▷ Eq. 40

  $u^{(k)} \leftarrow \exp(\Delta^{(k)}), \quad K3(k) \leftarrow u^{(k)} - \Delta^{(k)} - 1$ ▷ Eq. 41

end for

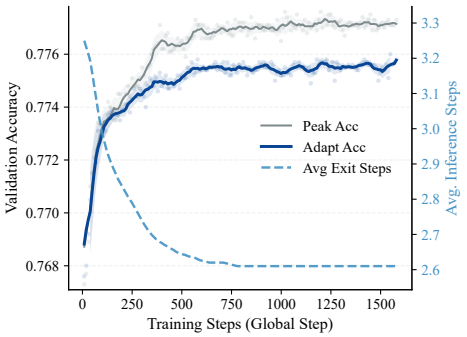
Let m_G be the rollout mask indicating the positions/steps evaluated in the rollout pass.

$L_{\text{KL}} \leftarrow \dfrac{\sum_{k=1}^K K3(k) \odot m_G}{\sum m_G + \varepsilon}$ ▷ Eq. 42

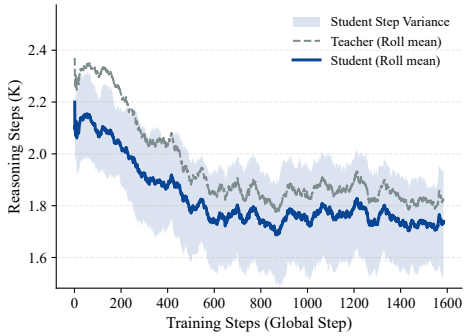
Total objective and updates

$L \leftarrow \alpha L_{\text{PG}} + \beta L_{\text{rep}} + \gamma L_{\text{ent}} + \delta L_{\text{KL}}$ ▷ Eq. 16

Update θ by one gradient step on L; update EMA teacher $\bar{\theta} \leftarrow \text{EMA}(\bar{\theta}, \theta)$.

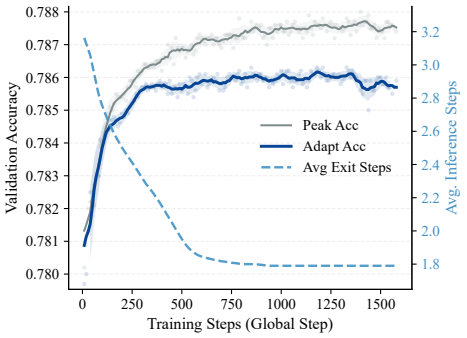


(a) **Validation accuracy–compute trade-off.** We report validation next-token accuracy under maximum-loop inference (Peak Acc) and adaptive early exit (Adapt Acc), together with the average exit steps (right y-axis). Faint dots indicate raw evaluation points and shaded bands reflect locally smoothed variability.

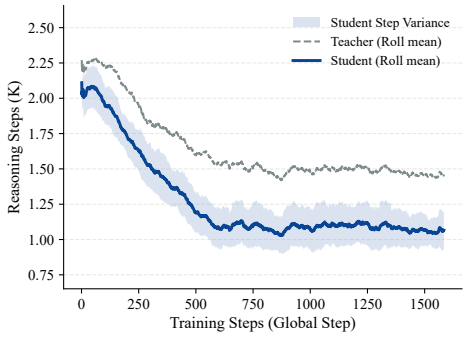


(b) **Evolution of early-exit behavior during training.** We plot the rolling-mean reasoning steps of the student (solid) and the EMA teacher reference (dashed); the shaded region shows the student's within-window variability. The student progressively aligns with the reference while converging to fewer steps, indicating improved early-exit calibration.

Figure 11: Training dynamics of LoopRPT on Ouro-1.4B.



(a) **Validation accuracy–compute trade-off.** We report validation next-token accuracy under maximum-loop inference (Peak Acc) and adaptive early exit (Adapt Acc), together with the average exit steps (right y-axis). Faint dots indicate raw evaluation points and shaded bands reflect locally smoothed variability.



(b) **Evolution of early-exit behavior during training.** We plot the rolling-mean reasoning steps of the student (solid) and the EMA teacher reference (dashed); the shaded region shows the student's within-window variability. The student progressively aligns with the reference while converging to fewer steps, indicating improved early-exit calibration.

Figure 12: Training dynamics of LoopRPT on Ouro-2.6B.

Case Analysis on General Tasks (MMLU-Pro)

Case 1: Engineering (Thermodynamics)
Question: Three kilograms of air are at an initial state of 100kPa, 300°K. The air is then compressed polytropically with $n = 2.56$, to a final pressure of 500kPa. Assuming constant specificheats, calculate the change in entropy using the three ideal gas equations.
Options: (A) 2.50 kJ/°K (B) 1.25 kJ/°K (C) 2.00 kJ/°K (D) 1.75 kJ/°K (E) 2.25 kJ/°K **(F) 1.57 kJ/°K** (G) 0.75 kJ/°K (H) 1.10 kJ/°K (I) 1.00 kJ/°K (J) 1.40 kJ/°K
Golden Answer: F

Base Model (Ouro-1.4B)	LoopRPT (Ours)
Full Reasoning: The change in entropy for a polytropic process is given by $\Delta S = nC_p \ln(T_2/T_1) - nR\ln(P_2/P_1)$. For air, $C_p = 1.005$ kJ/kg·K and $R = 0.287$ kJ/kg·K. Using the polytropic relation $P_1V_1^n = P_2V_2^n$, we find $V_2/V_1 = (P_1/P_2)^{1/n} \approx 0.554$. The temperature ratio $T_2/T_1 = (P_2/P_1)^{(n-1)/n} = (500/100)^{(1.56/2.56)} \approx 1.25$. Substituting into the entropy formula, $\Delta S \approx 3 \times 1.005 \times \ln(1.25) - 3 \times 0.287 \times \ln(5) \approx$ **0.75 kJ/°K**. **Conclusion:** The answer is (G) ✗	*Full Reasoning:* The change in entropy is given by $\Delta S = mC_p \ln\left(\frac{T_2}{T_1}\right) - mR\ln\left(\frac{P_2}{P_1}\right)$. For an ideal gas, $C_p = \frac{nR}{n-1}$ and R is the gas constant. Using the polytropic relation $P_1V_1^n = P_2V_2^n$, we find $T_2 = T_1\left(\frac{P_2}{P_1}\right)^{\frac{n-1}{n}}$. **Substituting the given values,** $T_2 = 300 \cdot \left(\frac{500}{100}\right)^{\frac{2.56-1}{2.56}} = 300 \cdot 5^{0.6094} \approx 799.6$ **K.** Then, $\Delta S = m\left[c_p \ln\left(\frac{T_2}{T_1}\right) - R\ln\left(\frac{P_2}{P_1}\right)\right]$. **Calculating each term,** $\Delta S = 3 \cdot \left[1.005\ln\left(\frac{799.6}{300}\right) - 0.287\ln(5)\right] \approx 1.57$ **kJ/K**. **Conclusion:** The answer is (F) ✓

Case 2: Biology (Thermoregulation)
Question: Which of the following would maintain a constant body temperature in spite of changes in the environmental temperature: frog, robin, fish, dog, or lizard?
Options: (A) Robin, Lizard (B) Frog, Fish (C) Lizard, Frog (D) Fish, Dog (E) Fish, Lizard (F) Dog, Frog (G) Lizard, Fish (H) Frog, Robin (I) Dog, Lizard **(J) Robin, Dog**
Golden Answer: J

Base Model (Ouro-1.4B)	LoopRPT (Ours)
Full Reasoning: The correct answer is (A) Robin, Lizard. **Birds and reptiles are ectotherms**, meaning they rely on external sources of heat to maintain their body temperature. Frogs and fish are also ectotherms, but they are not included in the correct answer. Dogs are endotherms, meaning they can regulate their body temperature internally. **Conclusion:** The answer is (A) ✗	*Full Reasoning:* The correct answer is (J). **Dogs and robins are endothermic**, meaning they can maintain a constant body temperature in spite of changes in the environmental temperature. **Frogs, fish, and lizards are ectothermic**, meaning they cannot maintain a constant body temperature in spite of changes in the environmental temperature. **Conclusion:** The answer is (J) ✓

Case 3: Computer Science (Discrete Logic)
Question: Which of the following expressions evaluates to true if the person is old enough to drive (≥ 16) but not old enough to vote (< 18)?
I. (age ≥ 16) AND (age ≤ 18) II. (age ≥ 16) AND (NOT(age ≥ 18)) III. (age < 18) AND (NOT(age < 16))
Options: (A) II only (B) I and II only (C) I and III only **(D) II and III only**
Golden Answer: D

Base Model (Ouro-1.4B)	LoopRPT (Ours)
Full Reasoning: The correct answer is II only. The expression (age ≥ 16) AND (NOT(age ≥ 18)) evaluates to true if the person is at least 16 years old but not at least 18 years old... **The other expressions do not correctly represent the conditions**. **Conclusion:** The answer is (A) ✗	*Full Reasoning:* The correct answer is (D). The first expression evaluates to true if the person is 16 or 17 years old, but it also evaluates to true if the person is 18 or older... The second expression evaluates to true if the person is 16 or 17 years old... **The third expression evaluates to true if the person is 16 or 17 years old, which is also what we want**. **Conclusion:** The answer is (D) ✓

Figure 13: **Comparative case study on general tasks (MMLU-Pro).** LoopRPT successfully identifies implicit logical equivalences (Case 3) and maintains factual consistency in biological classification (Case 2), whereas the base model exhibits knowledge hallucinations or incomplete reasoning.

Case Analysis on Mathematical Tasks (GSM8K)

Case 1: Scheduling and Time Management
Question: Doctor Jones is spending nine hours at the clinic. He has to do rounds to check on inpatients (20 min/inpatient) and has ten appointments (30 min each). How many hours will he have left if he has 9 inpatients?
Golden Answer: 1

Base Model (Ouro-1.4B)	LoopRPT (Ours)
Full Reasoning: Doctor Jones is spending 9 hours at the clinic. He has to do rounds to check on inpatients, which takes 20 minutes per inpatient. He has 10 appointments, which take 30 minutes each. **So in total, he spends 20 * 10 + 30 * 10 = 200 + 300 = 500 minutes**. Converting minutes to hours, 500 / 60 = 8.33 hours. So he has 9 - 8.33 = 0.67 hours left.	*Full Reasoning:* Doctor Jones is spending 9 hours at the clinic. **He has 9 inpatients at the clinic. So he spends 20 * 9 = 180 minutes doing rounds**. He has 10 appointments, which take 30 minutes each. So he spends 30 * 10 = 300 minutes on appointments. **In total, he spends 180 + 300 = 480 minutes**. 9 hours is 540 minutes. So he has 540 - 480 = 60 minutes left (1 hour).
Conclusion: The answer is 0.67 ✗	**Conclusion:** The answer is 1 ✓

Case 2: Multi-Agent Finance
Question: Maggie spent a quarter of her money, while Riza spent one-third of her money. They each had $60. How much money do the two of them have left?
Golden Answer: 85

Base Model (Ouro-1.4B)	LoopRPT (Ours)
Full Reasoning: Maggie spent 1/4 * 60 = 15. Riza spent 1/3 * 60 = 20. So in total they spent 15 + 20 = 35. **So they had 60 - 35 = 25**.	*Full Reasoning:* Maggie spent 1/4 * 60 = 15. Riza spent 1/3 * 60 = 20. Together, they spent 15 + 20 = 35. **They had 60 + 60 = 120. So they have 120 - 35 = 85 left**.
Conclusion: The answer is 25 ✗	**Conclusion:** The answer is 85 ✓

Case 3: Fractional Multi-step Calculation
Question: John has 8 classes (40 min each) for 5 days. He spends 1/16 of his weekly minutes each on Saturday and Sunday as extra time. How many hours a week does he spend learning?
Golden Answer: 30

Base Model (Ouro-1.4B)	LoopRPT (Ours)
Full Reasoning: Total class time is 8 * 40 * 5 = 1600 minutes. He spends 1/16 of his weekly minutes on Saturday and Sunday. **So in total he spends 1600 * 1/16 = 100 minutes**. So in total he spends 1600 + 100 = 1700 minutes (28.33 hours).	*Full Reasoning:* Total class time is 8 * 40 * 5 = 1600 minutes. **He then spends 1600 / 16 = 100 minutes each on Saturday and Sunday. So he spends 1600 + 100 + 100 = 1800 minutes in total**. 1800 minutes is 30 hours.
Conclusion: The answer is 28.33 ✗	**Conclusion:** The answer is 30 ✓

Figure 14: **Comparison of GSM8K mathematical reasoning**. Base model (Ouro-1.4B) frequently makes "set-neglect" errors (Case 2, 3) or hallucinations in variable mapping (Case 1), whereas LoopRPT correctly iterates through multi-step logic constraints.

Case Analysis on Coding Tasks (MBPP & HumanEval)

Case 1: MBPP (Mathematical Sequence)
Question: Write a function to find the nth octagonal number. (Assertion: `is_octagonal(5) == 65`)
Golden Answer: `return 3 * n * n - 2 * n`

Base Model (Ouro-1.4B)	LoopRPT (Ours)

Generated Code:
`def is_octagonal(n):`
`  # Returns True if n is octagonal...`
`  return n == (n * (3 * n - 1)) // 2`
	Generated Code:
	`def is_octagonal(n):`
	`  # Returns the n-th octagonal number.`
	`  return 3 * n * n - 2 * n`

Analysis: Confuses "nth number" request with "check property" and hallucinates pentagonal formula. **Conclusion:** Functional Failure ✗	**Analysis:** Correctly identifies the task and provides the exact octagonal sequence formula. **Conclusion:** Correct Generation ✓

Case 2: HumanEval (String Manipulation)
Question: Find the shortest palindrome that begins with a supplied string by reverse-prefixing.
Golden Answer: Iteratively find longest palindromic suffix and append reverse prefix.

Base Model (Ouro-1.4B)	LoopRPT (Ours)

Key Implementation:
`for i in range(len(string), 0, -1):`
`  if is_palindrome(string[-i:]):`
`  break`
`return string + string[:i-1][::-1]`
	Key Implementation:
	`for i in range(n):`
	`  if is_palindrome(string[i:]):`
	`  return string + string[:i][::-1]`

Analysis: Off-by-one error in indexing (`i-1`) leads to incorrect palindrome completion. **Conclusion:** Logic Error ✗	**Analysis:** Accurately iterates through split points to find the optimal palindromic suffix. **Conclusion:** Correct Generation ✓

Case 3: HumanEval (Cyclic Array Order)
Question: Determine if an array can be sorted by performing any number of right shifts.

Base Model (Ouro-1.4B)	LoopRPT (Ours)

Key Implementation:
`for i in range(n):`
`  if arr[i] > arr[(i + 1) % n]:`
`  return False`
`return True`
	Key Implementation:
	`sorted_arr = sorted(arr)`
	`for i in range(len(arr)):`
	`  if arr==sorted_arr[i:]+sorted_arr[:i]:`
	`  return True`

Analysis: Incorrectly implements a strict sorted check, failing for valid cyclic shifts. **Conclusion:** Logic Error ✗	**Analysis:** Correctly uses a cyclic comparison loop to verify all possible shift states. **Conclusion:** Correct Generation ✓

Figure 15: **Code generation comparison.** Base model (Ouro-1.4B) fails on boundary conditions and algorithmic interpretation, while LoopRPT demonstrates superior adherence to task constraints.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims made in the abstract and introduction are empirically verified in Section 5.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We mention our limitations in Appendix A.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: We provide the necessary justifications for the design of high-entropy token selection, noisy latent rollouts and EMA teacher in Appendix B.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide necessary implementation details in Appendix C- E. We do not release the source code.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: We do not provide the source code.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide training and testing details in Section 4 and Appendix E.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: All experimental results are averaged over three random seeds; we do not report error bars or significance tests.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide information about the computer resources used in Section 4 and Appendix E.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have read and understood the code of ethics and have made every effort to adhere to it.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: Our work contributes to a better understanding of the reinforcement learning for looped LMs. It does not impact society at large.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.

- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: We do not release any new models or datasets. Our work uses standard, publicly available resources and does not pose additional risk of misuse beyond their existing public availability.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We use publicly available datasets (OMNI-MATH) and pretrained models (Ouro), all of which are properly cited in the paper. We adhere to the licenses and terms of use as specified by their original creators.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.

- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We do not release new assets; an anonymous repository was made available only for the reviewing process.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: We didn't involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: We didn't involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: We used an LLM solely for editing assistance.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.