DECODING LOOPED TRANSFORMERS BETTER FOR (ALMOST) FREE

Anonymous authors
Paper under double-blind review

ABSTRACT

Looped Transformers achieve parameter efficiency by repeatedly executing a shared block across recurrent loops. Each loop yields an intermediate representation that is decodable for the same next token, yet standard decoding discards earlier states. Since earlier loops embody less computation, recurrence inherently supplies aligned weak-and-strong prediction pairs without auxiliary models or external training. We introduce **LoopCD**, a training-free contrastive decoding method that guides token selection by contrasting the final prediction with an earlier recurrent pass, operating either in logit space with one extra output pass (LoopCD-Logits) or in hidden-state space with zero output overhead (LoopCD-Hidden). Across four looped Transformer families, LoopCD delivers substantial, consistent gains at full recurrent depth: LoopCD-Logits raises Ouro-2.6B-Thinking's AIME 2024 pass@1 from 61.88% to 73.33%, while LoopCD-Hidden lifts Huginn's HumanEval pass@1 from 22.56% to 31.71%. Crucially, these performance gains enable halving the number of recurrent loops while still matching or exceeding full-depth unguided baselines, reducing forward FLOPs by 22.5% to 48.2%. By transforming intermediate recurrent states into effective guidance signals, LoopCD achieves superior decoding quality while substantially reducing inference compute.

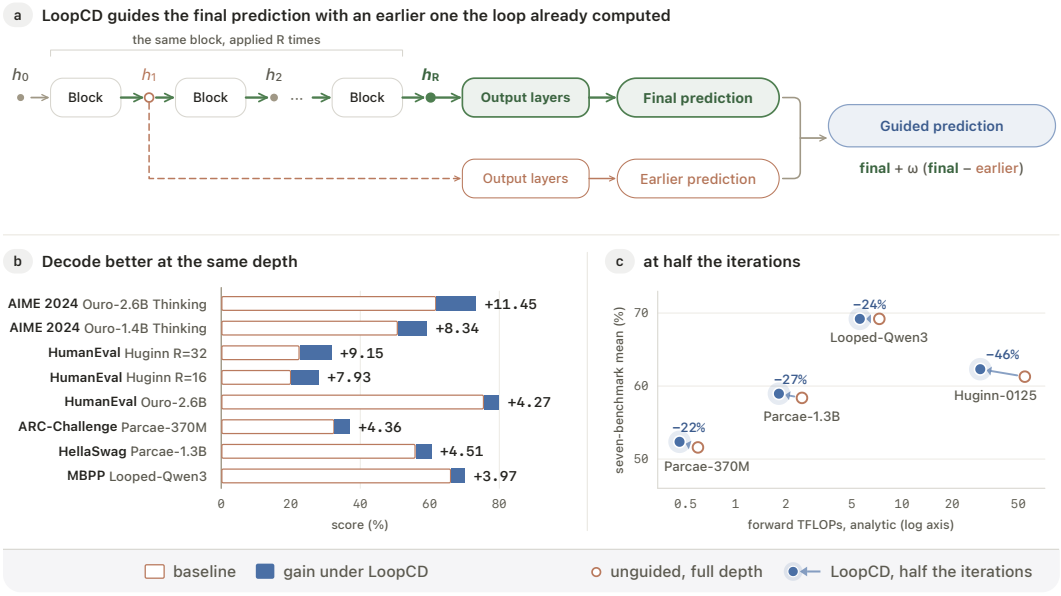

Figure 1: **Looped Transformers have already computed a weaker prediction on the way to the final one. LoopCD can leverage it to decode better at less or even zero extra computation.** **(a)** An earlier state, read through the same output layers, guides the final prediction, at one extra output pass for logits and none for hidden states. **(b)** Performance comparison of eight configurations at the baseline's recurrent depth. Huginn at $R = 32$ uses LoopCD-Hidden. **(c)** Each arrow runs from the unguided model at full depth (open) to LoopCD at half the iterations (filled), labeled with the FLOPs saved. All six settings are in Figure 4.

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

1 INTRODUCTION

Looped Transformers offer an appealing architecture for language modeling and reasoning by decoupling depth from parameter count: they repeatedly execute a block of shared layers, allowing effective computation to scale with recurrent iterations without increasing parameter count (Dehghani et al., 2019; Zhu et al., 2026; Geiping et al., 2025; Prairie et al., 2026). During generation, the model refines its representation across several recurrent passes before emitting a token. Existing methods focus exclusively on the state emitted by the final iteration. However, each intermediate pass also produces a hidden state that can be decoded into a valid token distribution, which standard decoding typically discards.

Contrastive decoding guides a strong predictor away from an aligned weaker reference (Li et al., 2023). Similar principles appear in diffusion models, where autoguidance improves sample fidelity by contrasting against a degraded version of the same network (Karras et al., 2024). In standard language models, obtaining an aligned weak predictor typically requires an auxiliary smaller model, a perturbed context, or an intermediate layer (Li et al., 2023; Shi et al., 2024; Chuang et al., 2024). In contrast, since a looped Transformer repeatedly executes a shared parameter block over depth, each iteration adds compute while operating in the same representation space. The recurrent trajectory thus inherently produces an aligned weak-to-strong prediction pair for the same prefix, without auxiliary models or modified prompts.

We introduce **LoopCD**, which turns this weak-to-strong trajectory into a direct inference-time decoding method. LoopCD contrasts the final prediction with the first iteration to guide token selection along the direction of recurrent refinement, requiring no additional training and no extra recurrent loops. We present two variants of this method: **LoopCD-Logits**, which evaluates both states through the output layers with one extra output pass, and **LoopCD-Hidden**, which combines the representations in hidden-state space before the output layers to preserve a single output pass with no extra overhead. Both variants can use a fixed guidance strength, and LoopCD-Logits can also adapt it per token to the uncertainty of the prediction.

Our experiments demonstrate that LoopCD provides substantial, consistent gains across two distinct computational regimes. At full recurrent depth, LoopCD delivers improvements across all four evaluated model families: Ouro (Zhu et al., 2026), Huginn (Geiping et al., 2025), Parcae (Prairie et al., 2026), and Looped-Qwen3 (Chen et al., 2026). The gains span mathematical reasoning (AIME, OlympiadBench), code generation (HumanEval, MBPP), and multiple-choice benchmarks (MMLU, ARC). LoopCD-Logits lifts Ouro-2.6B-Thinking's (Zhu et al., 2026) AIME 2024 pass@1 from 61.88% to 73.33% (Figure 1), while in hidden space, LoopCD-Hidden achieves similar gains with zero output overhead, raising Huginn's HumanEval pass@1 from 22.56% to 31.71%. Crucially, these substantial performance gains enable reducing the number of recurrent iterations: operating at only half the recurrent depth, LoopCD still matches or exceeds the unguided full-depth baseline across evaluated model families, reducing forward FLOPs by 22.5% to 48.2%.

Summary of contributions. This work makes three primary contributions. First, building on the observation that looped Transformers inherently generate aligned weak-and-strong representations across recurrent depth, we propose **LoopCD**, a training-free contrastive decoding method requiring no auxiliary models or external training. Second, we establish two guidance spaces, showing that LoopCD-Hidden combines representations before the coda layers to eliminate extra output projection overhead while preserving accuracy gains. Third, we demonstrate across four model families that LoopCD consistently improves reasoning and generation at full recurrent depth, and that these substantial gains enable halving recurrent iterations while matching or exceeding full-depth baseline accuracy, reducing forward FLOPs by 22.5% to 48.2%.

2 PRELIMINARIES: LOOPED TRANSFORMERS

A looped Transformer updates its hidden representations by repeatedly applying a block with shared parameters (Dehghani et al., 2019). Each application is a *recurrent iteration*: increasing the iteration count increases effective depth and computation while keeping the stored parameters fixed.

Looped Transformer architectures vary in how they integrate the shared block within the overall network (Figure 2). Ouro repeats its full decoder stack (Zhu et al., 2026). Huginn places the recurrent

block between input-processing layers, called the *prelude*, and post-loop layers, called the *coda*. Its updates also receive a fixed representation from the prelude (Geiping et al., 2025). Parcae and Looped-Qwen3 also retain layers before and after the recurrent block, although their update mechanisms differ: Parcae learns its recurrent dynamics, while Looped-Qwen3 applies damped updates to a frozen middle-layer window (Prairie et al., 2026; Chen et al., 2026). Appendix A.2 provides architectural details for each evaluated model family.

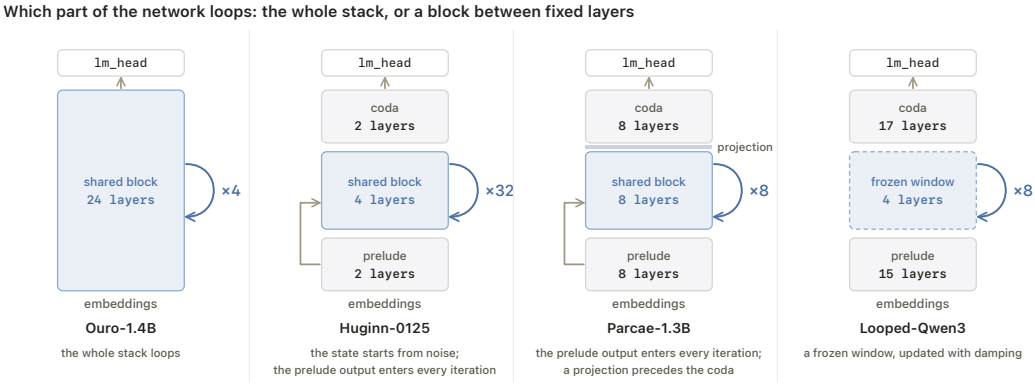

Figure 2: **Looped Transformers differ in which part of the network loops.** Ouro repeats its full stack, while Huginn, Parcae, and Looped-Qwen3 repeat a shared block (blue) between fixed prelude and coda layers (gray). Layer and iteration counts are for Ouro-1.4B, Huginn-0125, Parcae-1.3B, and Looped-Qwen3 (Table A1).

For a fixed input prefix, let h_0 be the initial state of the recurrent block and h_k be its state after k iterations. The block applies an update f for R iterations, after which the coda layers and language modeling head map the final state h_R to token logits:

$$\begin{aligned} h_0 \xrightarrow{f} h_1 \xrightarrow{f} \cdots \xrightarrow{f} h_R, \\ z_R = \texttt{lm_head}\big(\texttt{coda}(h_R)\big). \end{aligned} \tag{1}$$

Here, `lm_head` consists of the final layer normalization and vocabulary projection. Standard autoregressive decoding samples the next token from $p_R = \mathrm{softmax}(z_R)$.

3 LoopCD: Contrastive Decoding from Recurrent States

LoopCD typically uses the first recurrent state h_1 (Eq. (1)) as a weak reference for the final state $\boldsymbol{h_R}$ (Section 5.1). While standard decoding relies solely on h_R (the *unguided* baseline), LoopCD produces a *guided* prediction by extrapolating away from h_1 in either hidden-state or logit space. This contrastive update (Li et al., 2023) shares the principle of autoguidance and classifier-free guidance in diffusion models (Ho & Salimans, 2021; Karras et al., 2024) by steering predictions away from a degraded or unconditional reference. LoopCD entails two design choices: guidance space (Section 3.1) and strength ω (Section 3.2).

3.1 Guidance Space

LoopCD applies the contrast before the coda layers or after the language modeling head (Figure 3).

LoopCD-Hidden. LoopCD combines the reference and final hidden states before the coda layers:

$$h' = \boldsymbol{h_R} + \omega\left(\boldsymbol{h_R} - h_1\right), \qquad \omega \geq 0. \tag{2}$$

The coefficient ω sets the strength of the contrast, with $\omega = 0$ recovering the unguided state h_R. The combined state then passes through the coda layers and language modeling head: $z' = \texttt{lm_head}(\texttt{coda}(h'))$. The coda layers and `lm_head` each execute once, preserving the baseline's single pass through these layers (Section 4.3).

LoopCD-Logits. LoopCD contrasts the logits obtained from the reference and final states after the coda layers and language modeling head:

$$z_1 = \texttt{lm_head}\big(\texttt{coda}(h_1)\big), \qquad \boldsymbol{z_R} = \texttt{lm_head}\big(\texttt{coda}(\boldsymbol{h_R})\big),$$
$$z' = \boldsymbol{z_R} + \omega\,\big(\boldsymbol{z_R} - z_1\big), \qquad \omega \geq 0. \tag{3}$$

Computing the reference logits z_1 requires one additional pass through the coda layers and `lm_head`. Appendix A.3 gives the equivalent update in probability space. Both forms decode the next token from $\mathrm{softmax}(z')$.

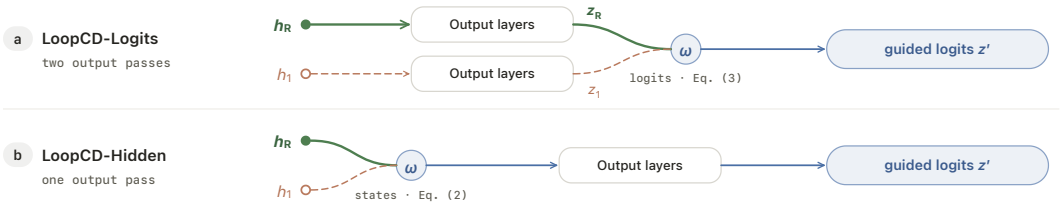

Figure 3: **LoopCD applies the contrast before or after the coda and language modeling head.** LoopCD-Hidden combines h_1 and h_R (Eq. (2)) and evaluates the coda and `lm_head` once, while LoopCD-Logits evaluates both states through them and combines the logits (Eq. (3)), in two passes. Colors follow the equations: h_1 is the reference, $\boldsymbol{h_R}$ the final state, and z' the guided result.

3.2 Guidance Strength

The coefficient ω controls how far guidance extrapolates beyond the unguided prediction.

Fixed strength. A constant ω is used for every token within a given evaluation setting (Appendix B.4).

Adaptive strength. LoopCD can adjust ω at each token using the uncertainty of the unguided prediction. Let $p_{R,(1)}$ and $p_{R,(2)}$ be the two largest token probabilities in $p_R = \mathrm{softmax}(z_R)$. We set

$$\omega = \omega_{\max}\big[1 - (p_{R,(1)} - p_{R,(2)})\big], \tag{4}$$

where $\omega_{\max}$ is the maximum strength. The rule approaches this maximum as the two leading probabilities become equal and reduces guidance as one candidate becomes dominant. The probability margin measures the uncertainty between the leading candidates directly, without computing entropy over the full vocabulary.

LoopCD-Logits already computes the unguided logits z_R needed by the adaptive rule. Applying the same rule to LoopCD-Hidden would require a coda and head pass on h_R to obtain p_R, followed by another pass on h'. We therefore use fixed and adaptive strength for LoopCD-Logits, and fixed strength for LoopCD-Hidden.

4 Experiments

Setup. We evaluate Ouro (Zhu et al., 2026), Huginn (Geiping et al., 2025), Parcae (Prairie et al., 2026), and Looped-Qwen3 (Chen et al., 2026), built from Qwen3-4B (Yang et al., 2025). Guided and unguided runs share checkpoints and prompts, and full-depth comparisons also match recurrent depth. Appendix B.1 details model configurations and paired comparisons, and Appendix B.2 specifies the benchmark protocols.

4.1 Gains on Mathematical Reasoning and Code Generation

LoopCD substantially improves mathematical reasoning pass rates. Both Ouro-Thinking models improve pass@1 and pass@10 on AIME 2024 (HuggingFaceH4, 2025), AIME 2025 (OpenCompass, 2025), and OlympiadBench (He et al., 2024) at the same recurrent depth as their unguided baselines (Table 1(a)). For Ouro-2.6B-Thinking, adaptive LoopCD-Logits raises AIME 2024 pass@1

from 61.88% to 73.33%, AIME 2025 pass@1 from 49.58% to 56.88%, and OlympiadBench pass@1 from 64.05% to 67.29%. Fixed and adaptive guidance rules both improve all evaluated benchmarks across pass@1 (average gains of 5.27 to 7.33 points) and pass@10 (average gains of 2.53 to 4.12 points), with the adaptive rule achieving the highest pass@1 gain for both models.

On code generation, LoopCD consistently improves execution pass rates across model families and scales. We evaluate HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021) with base and extended tests from EvalPlus (Liu et al., 2023) (Table 1(b)). For Huginn-0125 at $R = 32$, adaptive LoopCD-Logits lifts HumanEval pass@1 from 23.17% to 28.66%, and at $R = 16$ from 20.12% to 28.05%. Combining the two hidden states before the output layers achieves even stronger results: at $R = 32$, LoopCD-Hidden raises HumanEval pass@1 from its own baseline of 22.56% to 31.71%, and at $R = 16$ matches the adaptive logit score, moving from 21.95% to 28.05% (Table 3(b)).

(a) Mathematical reasoning

	AIME 2024		AIME 2025		OlympiadBench		Mean Δ	
Model	pass@1	pass@10	pass@1	pass@10	pass@1	pass@10	pass@1	pass@10
Ouro-1.4B-Thinking	50.83	80.33	38.96	68.09	61.27	76.69		
LoopCD, fixed	$56.88_{+6.05}$	$84.82_{+4.49}$	$46.25_{+7.29}$	$69.23_{+1.14}$	$63.75_{+2.48}$	$78.66_{+1.97}$	+5.27	+2.53
LoopCD, adaptive	$59.17_{+8.34}$	$88.19_{+7.86}$	$47.08_{+8.12}$	$70.30_{+2.21}$	$63.48_{+2.21}$	$78.99_{+2.30}$	**+6.22**	**+4.12**
Ouro-2.6B-Thinking	61.88	82.65	49.58	75.95	64.05	78.22		
LoopCD, fixed	$70.62_{+8.74}$	$88.33_{+5.68}$	$58.33_{+8.75}$	$81.21_{+5.26}$	$66.66_{+2.61}$	$79.52_{+1.30}$	+6.70	**+4.08**
LoopCD, adaptive	$73.33_{+11.45}$	$88.33_{+5.68}$	$56.88_{+7.30}$	$79.53_{+3.58}$	$67.29_{+3.24}$	$80.08_{+1.86}$	**+7.33**	+3.71

(b) Code generation

	Unguided				LoopCD, fixed					LoopCD, adaptive				
	HumanEval		MBPP		HumanEval		MBPP			HumanEval		MBPP		
Model	Base	Ext.	Base	Ext.	Base	Ext.	Base	Ext.	Mean Δ	Base	Ext.	Base	Ext.	Mean Δ
Ouro-1.4B	69.51	65.85	73.28	61.11	71.95	66.46	72.75	60.05	+0.37	72.56	67.07	74.07	61.11	**+1.27**
Ouro-2.6B	75.61	70.12	77.78	64.02	77.44	71.34	78.31	64.81	+1.09	79.88	72.56	79.10	66.14	**+2.54**
Huginn-0125, R=32	23.17	20.12	42.06	32.54	27.44	23.78	44.97	37.57	**+3.97**	28.66	24.39	41.27	33.86	+2.57
Huginn-0125, R=16	20.12	17.68	40.74	32.01	26.83	20.73	41.01	34.13	+3.04	28.05	25.61	39.68	32.80	**+3.90**
Looped-Qwen3	79.88	71.95	66.14	56.08	79.27	71.34	70.11	58.99	**+1.42**	80.49	73.78	67.99	57.41	+1.41

Table 1: **LoopCD improves mathematical reasoning and code generation at full recurrent depth.** Panel (a): Ouro-Thinking, pass@1 and pass@10 from sixteen samples per problem. Panel (b): five configurations, base and extended tests. Baselines in gray. Subscripts in (a) give the change from the baseline. In (b), a guided score is green above its baseline and red below. Bold marks the better rule's Mean Δ (Appendix B.2).

4.2 Consistent Improvements Across Architectures and Suites

LoopCD improves average multiple-choice accuracy across all four model families. We score candidate answers by likelihood on seven benchmarks: ARC-Challenge and ARC-Easy (Clark et al., 2018), SciQ (Welbl et al., 2017), MMLU (Hendrycks et al., 2021), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2020), and PIQA (Bisk et al., 2020). Every evaluated model improves its benchmark mean under LoopCD-Logits, with the adaptive strength method leading across nearly all configurations (Table 2).

LoopCD is robust against different training methods: while Ouro supervises readouts after every pass, Huginn and Parcae sample recurrent depths during pre-training, and Looped-Qwen3 retrofits recurrence onto a frozen non-recurrent model. LoopCD delivers consistent improvements across all four regimes, demonstrating that its benefit does not depend on a specific recurrent training objective. These improvements persist across diverse prompting formats, from zero-shot evaluation on Huginn to twenty-five demonstrations on Ouro, as well as under generative chain-of-thought decoding on MMLU-Pro (Wang et al., 2024) (Table A6).

Model	ARC-C	ARC-E	SciQ	MMLU	HellaSwag	WinoGrande	PIQA	Mean Δ
Ouro-1.4B	60.41	84.09	94.70	44.45	74.48	70.80	78.51	–
LoopCD, fixed	62.29$_{+1.88}$	84.39$_{+0.30}$	95.30$_{+0.60}$	45.35$_{+0.90}$	75.30$_{+0.82}$	69.85$_{-0.95}$	79.05$_{+0.54}$	**+0.58**
LoopCD, adaptive	61.69$_{+1.28}$	84.26$_{+0.17}$	95.20$_{+0.50}$	45.67$_{+1.22}$	75.57$_{+1.09}$	68.35$_{-2.45}$	78.73$_{+0.22}$	+0.29
Ouro-2.6B	66.21	88.01	94.20	51.09	79.52	77.66	80.25	–
LoopCD, fixed	68.60$_{+2.39}$	89.10$_{+1.09}$	94.60$_{+0.40}$	51.82$_{+0.73}$	81.07$_{+1.55}$	76.87$_{-0.79}$	80.74$_{+0.49}$	+0.84
LoopCD, adaptive	69.71$_{+3.50}$	89.02$_{+1.01}$	94.30$_{+0.10}$	52.30$_{+1.21}$	81.66$_{+2.14}$	75.93$_{-1.73}$	80.69$_{+0.44}$	**+0.95**
Huginn-0125 R=32	37.63	64.31	90.40	33.47	66.65	60.77	75.79	–
LoopCD, fixed	39.51$_{+1.88}$	64.10$_{-0.21}$	91.10$_{+0.70}$	33.57$_{+0.10}$	69.00$_{+2.35}$	61.96$_{+1.19}$	74.97$_{-0.82}$	+0.74
LoopCD, adaptive	39.42$_{+1.79}$	64.69$_{+0.38}$	91.20$_{+0.80}$	33.61$_{+0.14}$	69.03$_{+2.38}$	61.33$_{+0.56}$	75.79$_{+0.00}$	**+0.86**
Huginn-0125 R=16	37.63	64.31	90.30	33.51	66.28	60.30	75.52	–
LoopCD, fixed	39.76$_{+2.13}$	64.86$_{+0.55}$	90.90$_{+0.60}$	33.48$_{-0.03}$	68.90$_{+2.62}$	59.75$_{-0.55}$	74.54$_{-0.98}$	+0.62
LoopCD, adaptive	39.33$_{+1.70}$	64.94$_{+0.63}$	91.20$_{+0.90}$	33.46$_{-0.05}$	68.82$_{+2.54}$	59.67$_{-0.63}$	74.86$_{-0.66}$	**+0.63**
Parcae-370M	32.59	54.42	78.60	29.34	44.10	55.09	66.97	–
LoopCD, fixed	35.15$_{+2.56}$	55.05$_{+0.63}$	79.40$_{+0.80}$	29.58$_{+0.24}$	46.66$_{+2.56}$	53.59$_{-1.50}$	68.01$_{+1.04}$	+0.90
LoopCD, adaptive	36.95$_{+4.36}$	54.63$_{+0.21}$	79.80$_{+1.20}$	29.63$_{+0.29}$	47.86$_{+3.76}$	53.83$_{-1.26}$	67.41$_{+0.44}$	**+1.29**
Parcae-1.3B	41.21	65.07	83.70	32.63	55.93	57.85	72.03	–
LoopCD, fixed	43.86$_{+2.65}$	65.91$_{+0.84}$	85.80$_{+2.10}$	33.19$_{+0.56}$	59.19$_{+3.26}$	58.72$_{+0.87}$	71.44$_{-0.59}$	+1.38
LoopCD, adaptive	45.31$_{+4.10}$	66.16$_{+1.09}$	85.30$_{+1.60}$	33.52$_{+0.89}$	60.44$_{+4.51}$	58.48$_{+0.63}$	70.35$_{-1.68}$	**+1.59**
Looped-Qwen3	62.12	79.25	95.30	47.97	69.14	65.90	75.19	–
LoopCD, fixed	62.54$_{+0.42}$	79.00$_{-0.25}$	95.00$_{-0.30}$	48.81$_{+0.84}$	71.39$_{+2.25}$	65.75$_{-0.15}$	74.65$_{-0.54}$	+0.32
LoopCD, adaptive	64.08$_{+1.96}$	78.58$_{-0.67}$	95.10$_{-0.20}$	49.07$_{+1.10}$	72.31$_{+3.17}$	66.14$_{+0.24}$	75.14$_{-0.05}$	**+0.79**

Table 2: **Every logit configuration improves its seven-benchmark mean under LoopCD.** Candidates scored by likelihood. Baselines in gray, then LoopCD at fixed and adaptive strength. Subscripts give the change from the baseline, and bold marks the better rule's mean (Appendix B.2).

LoopCD-Hidden provides zero-overhead improvements. Table 3 evaluates the hidden-state form at full recurrent depth. The two states are combined before the output layers, so guidance adds no extra forward computation (Section 3.1). In multiple-choice scoring (Table 3(a)), the seven-benchmark mean rises by +0.83 and +0.75 points on Huginn at $R = 32$ and $R = 16$, exceeding fixed LoopCD-Logits gains (+0.74 and +0.62, Table 2), and by +0.85 points on Parcae-1.3B. In code generation (Table 3(b)), the four-column mean improves by +5.13 points at $R = 32$, ahead of both logit rules (+3.97 and +2.57, Table 1(b)), and by +3.76 at $R = 16$, with every code column improving at both depths. Appendix D.3 analyzes how much of the logit gain the hidden form retains behind a deeper coda.

(a) Multiple-choice scoring

Benchmark	Huginn-0125 R=32		Huginn-0125 R=16		Parcae-1.3B R=8	
ARC-C	37.97	38.74$_{+0.77}$	37.37	37.97$_{+0.60}$	40.36	42.66$_{+2.30}$
ARC-E	64.18	64.52$_{+0.34}$	64.14	64.35$_{+0.21}$	64.60	65.36$_{+0.76}$
HellaSwag	66.56	68.86$_{+2.30}$	66.32	68.16$_{+1.84}$	55.99	58.19$_{+2.20}$
WinoGrande	61.17	62.35$_{+1.18}$	61.01	62.83$_{+1.82}$	58.96	59.12$_{+0.16}$
PIQA	75.52	75.57$_{+0.05}$	75.19	75.08$_{-0.11}$	72.63	71.11$_{-1.52}$
SciQ	90.30	91.00$_{+0.70}$	90.00	90.70$_{+0.70}$	83.80	85.30$_{+1.50}$
MMLU	33.58	34.08$_{+0.50}$	33.54	33.74$_{+0.20}$	32.48	33.06$_{+0.58}$
Mean Δ	–	+0.83	–	+0.75	–	+0.85

(b) Code generation

	Huginn-0125 R=32		Huginn-0125 R=16	
HumanEval				
Base	22.56	31.71	21.95	28.05
Ext.	19.51	29.27	18.29	23.78
MBPP				
Base	40.74	41.27	41.53	42.86
Ext.	33.07	34.13	33.07	35.19
Mean Δ	–	+5.13	–	+3.76

Table 3: **LoopCD-Hidden: Guidance in hidden-state space improves performance across configurations with no added output overhead.** Fixed strength and one output pass (references in Appendix B.4). Baselines are in gray, measured under the hidden-state runs' own prompts. A configuration is reported where its baseline agrees with the logit row's within two points (Appendix B.2).

4.3 Fewer Iterations, Matching or Superior Performance

LoopCD can reduce the number of loop iterations while matching or surpassing baseline models at full depth. Given the performance gains at fixed depth, we investigate whether guidance can substitute for recurrent iterations to reduce inference compute. Reducing recurrent iterations by half incurs an accuracy penalty of 0.17 to 1.29 points on the unguided seven-benchmark multiple-choice mean. Applying LoopCD at this halved depth adds 0.75 to 1.29 points, closing the deficit across all six evaluated settings spanning Huginn, Parcae, and Looped-Qwen3 (Figure 4). In particular, Huginn-0125 evaluated at sixteen of its thirty-two iterations outperforms its full-depth unguided baseline by 1.02 points under LoopCD-Logits and 0.51 points under LoopCD-Hidden. Looped-Qwen3 matches its full-depth baseline at half iterations under LoopCD-Hidden (+0.00 points).

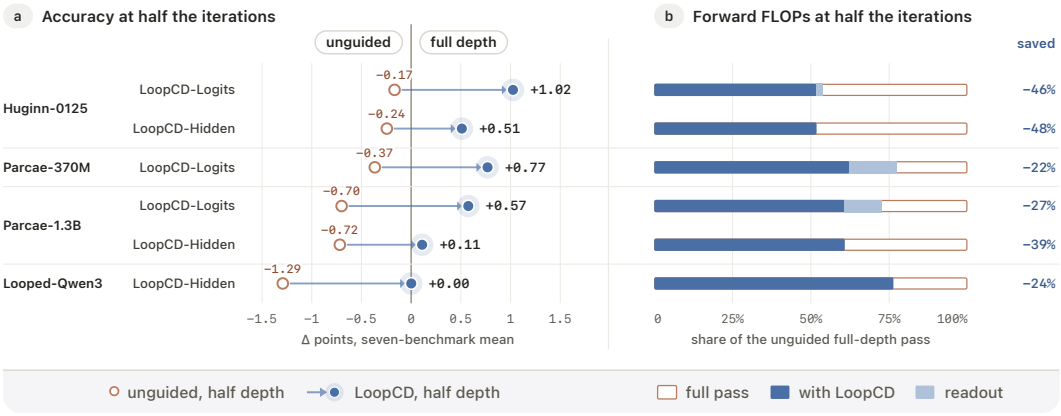

Figure 4: **At half the recurrent iterations, LoopCD matches or exceeds the unguided model at full depth, for a fifth to a half fewer FLOPs.** **(a)** Change in the seven-benchmark mean from the unguided model at full depth: the unguided model at half depth (open) and LoopCD at half depth (filled). **(b)** Forward FLOPs at half depth (solid) as a share of the unguided full-depth pass (outlined). The lighter segment is the readout of logit guidance. Looped-Qwen3 exits after four of eight substeps. Settings are in Table A9.

LoopCD shifts the accuracy-compute Pareto frontier, saving up to half the forward FLOPs at equal accuracy. Halving the iterations removes the loop's share of the forward pass, and LoopCD-Logits adds one pass through the coda layers and language modeling head. With all guidance overhead counted, the guided model at half depth needs 0.52 to 0.78 of the unguided full-depth FLOPs (Figure 4(b)), saving 22.5% to 48.2% at equal or higher accuracy.

5 How and Why LoopCD Works

Two findings explain the gains in Section 4. First, the recurrence already supplies a weak and a strong predictor, and the first iteration, which disagrees most with the final prediction, is the most effective logit reference (Section 5.1). Second, the contrast works mainly by re-ranking the leading candidates, and its gains come from close decisions on which the unguided model is uncertain (Section 5.2). Appendices E and F extend these two analyses. Appendix G sweeps the guidance strength: multiple-choice scoring peaks near $\omega = 0.5$ over a broad range, generation needs ω between 0.2 and 0.3, and the adaptive rule keeps a positive gain at caps where a fixed strength overshoots.

5.1 Built-in Weak and Strong Predictions Across Recurrence

The recurrent trajectory contains a weak and a strong predictor. Read through the same output layers, each intermediate state is already a prediction from the same input, so LoopCD takes both predictors from the recurrence the model already runs. Evaluated alone, the first iteration's prediction trails the final one on the seven-benchmark mean by 4.0 points for Parcae-370M up to 23.7 for Huginn, whose recurrence starts from noise, and accuracy rises across the iterations (Appendix E.1, Figure A5).

The gain follows how much the reference disagrees with the final prediction. What matters is not how weak the reference is but how far its prediction is from the final one (Appendix E.1). Recurrence changes the prediction a lot in its first iterations and little afterwards (Figure 5(a, b)), so a later reference sits closer to the final prediction and leaves less to extrapolate. On ARC-Challenge, Huginn's first step picks a different option from the final prediction on 51% of the questions and adds 3.07 points at $\omega = 0.5$, while its sixteenth step disagrees on only 7% and adds 0.17. The same pattern holds across four models (Figure 5(c)). The reference supplies a direction, and a reference that has already converged supplies almost none.

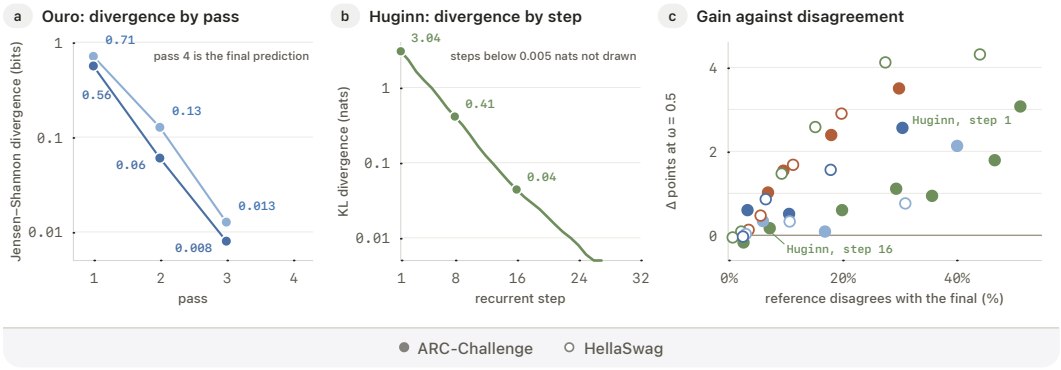

Figure 5: **The disagreement between the reference and the final prediction shrinks with depth, and the gain follows it.** **(a)** Jensen–Shannon divergence (bits) between Ouro's prediction at each pass and its final one, ARC-Challenge. **(b)** Kullback–Leibler divergence (nats) between Huginn's prediction at each step and its final one. **(c)** The gain of LoopCD-Logits at $\omega = 0.5$ against the share of questions on which the reference alone picks a different option from the final prediction, four models, ARC-Challenge (filled) and HellaSwag (open).

The first iteration is the best logit reference, but Huginn's hidden-state reference must come after its burn-in. For every swept model, LoopCD-Logits gains most with the first iteration as reference, and the gain shrinks as the reference moves deeper (Figure 6(a)). LoopCD-Hidden behaves differently on Huginn, whose recurrence starts from Gaussian noise. Its first steps, the burn-in, mostly remove that noise. The coda layers project the noise away when the reference is read out as logits, but a hidden-state contrast carries it into the update. Huginn's first hidden state therefore yields -0.58 points, and the gain peaks at $+0.83$ with the sixth (Figure 6(b)). Parcae and Looped-Qwen3 start from deterministic states and use their first state in both guidance spaces (Appendix B.4).

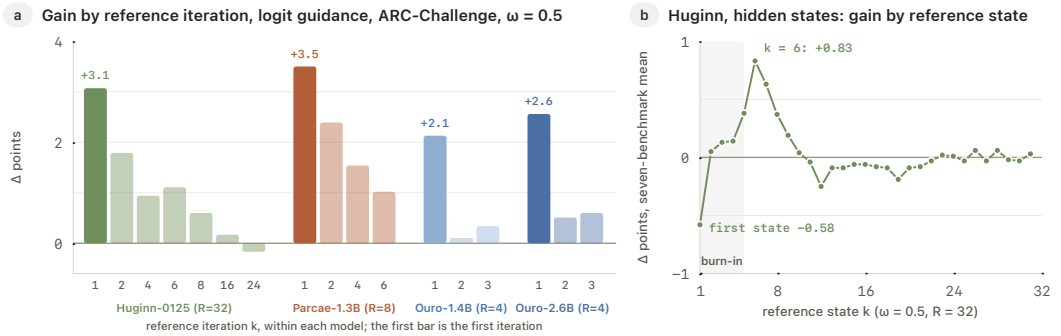

Figure 6: **The first iteration is the best logit reference; Huginn's hidden-state reference comes after its burn-in.** **(a)** The gain of LoopCD-Logits at $\omega = 0.5$ on ARC-Challenge by reference iteration k, grouped by model, the first iteration in full colour. **(b)** Huginn at $R = 32$, LoopCD-Hidden at $\omega = 0.5$: seven-benchmark mean change by reference state, burn-in shaded.

5.2 Re-ranking Close Decisions Under Uncertainty

The contrast vector decomposes into parallel temperature scaling and orthogonal re-ranking, with the re-ranking component driving performance gains. Projecting the logit contrast $z_R - z_1$ into components parallel and orthogonal to z_R (Figure 7(a)) separates their functional roles. The parallel component rescales logits uniformly, acting as a temperature adjustment that preserves token order. The orthogonal component alters relative token distances, acting as a pure re-ranking that directly changes top-token selection. On HellaSwag, applying the orthogonal re-ranking component alone achieves gains matching or exceeding the full update up to $\omega = 1.5$, yielding +1.72 points against +0.76 for Ouro-1.4B and +2.27 against +1.56 for Ouro-2.6B at $\omega = 0.5$ (Figure 7(b)). LoopCD-Hidden makes the same re-ranking decisions in representation space but has no temperature part, and on HellaSwag it gains +1.73 points for Ouro-1.4B, matching the re-ranking part alone and above the logit form (Appendix F).

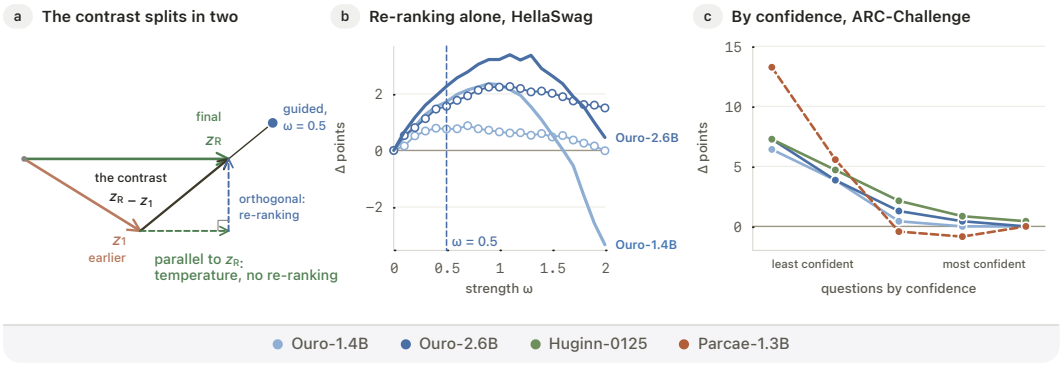

Figure 7: **The re-ranking carries the gain, and the gain lands where the model is uncertain. (a)** An illustrative contrast $z_R - z_1$ splits into a part parallel to z_R, which rescales the logits like a temperature, and an orthogonal part, which re-ranks tokens. **(b)** HellaSwag, the two Ouro models: accuracy change as the strength grows, re-ranking part alone (thick) and full update (thin). **(c)** ARC-Challenge questions in five equal groups by the unguided prediction's confidence, least confident first: accuracy change per group, LoopCD-Logits at $\omega = 0.5$, four models.

Guidance helps most where the model is uncertain. A re-ranking can only change a decision it can reach. The update flips an answer only when the push it adds exceeds the gap between the final prediction's two best options, so confident decisions stay as they are and the undecided ones move. Grouping ARC-Challenge questions by that gap, guidance at $\omega = 0.5$ adds between +6.4 and +13.3 points on the least confident fifth of questions for four models and at most +0.4 on the most confident fifth (Figure 7(c); Appendix F.1). Overall, relatively few answers flip (between 5.8% and 14.9%), and the net gain of 2.1 to 3.5 points comes mainly from resolving close decisions. This dynamic explains why the adaptive rule of Section 3.2 focuses guidance strength where the probability margin is narrow, concentrating updates where they can act.

6 Conclusion

Looped Transformers naturally compute aligned weak and strong predictions across recurrent depths, yet standard decoding discards these intermediate states. LoopCD repurposes them, explicitly extrapolating the token preference shifts induced by recurrence at the cost of only one extra output pass for LoopCD-Logits, or even zero additional compute for LoopCD-Hidden. At a fixed recurrent depth, this intrinsic contrastive signal raises Ouro's mean mathematical reasoning pass@1 by five to seven points and systematically improves performance across four distinct model families. Furthermore, at half the standard iteration budget, LoopCD matches or exceeds full-depth baseline accuracy, yielding a 22.5% to 48.2% reduction in forward FLOPs. These gains stem from a straightforward mechanism: extrapolating the model's belief revision trajectory decisively re-ranks uncertain boundary decisions toward the options favored by the recurrent dynamics. Models trained with explicit intermediate supervision, such as Ouro, may render this contrastive pair even more effective.

AI USE STATEMENT

In this work, we used generative AI tools to draft sections of the paper and to aid or polish the writing. We have not used generative AI tools for retrieval and discovery, research ideation or execution, generating synthetic datasets, or proving mathematical claims, and the rest of the required disclosure tasks are not applicable to this work. We have reviewed all AI-assisted work. Specifically, all drafted and polished text was manually read, rigorously edited by the authors, and cross-checked against our empirical findings to ensure scientific accuracy. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

LoopCD changes only how an existing model's predictions are decoded at inference time. It trains no model, collects no data, and involves no human subjects. All experiments use checkpoints and benchmarks released in prior work (Section 4 and Appendix B.2). We did not evaluate whether guidance changes the safety or fairness behavior of the underlying models.

REPRODUCIBILITY STATEMENT

Section 3.1 defines both forms of LoopCD (Equations 2 and 3), Section 3.2 gives the adaptive strength rule (Equation 4), and Appendix A.3 derives the probability form of LoopCD-Logits. Appendix A.2 and Table A1 describe each evaluated architecture and the states LoopCD reads from it. Appendix B specifies the evaluation: the configurations and the pairing of each guided run with its unguided baseline (Appendix B.1 and Table A2), the benchmarks, harnesses, answer extractors, and metrics (Appendix B.2), the prompts and shot counts (Appendix B.3 and Tables A3 and A4), and the selected strengths and reference states together with the sweeps that selected them (Appendix B.4, Table A5, and Appendix G.1). Table A9 lists the settings of the reduced-depth comparisons, and Appendix D.1 gives how every FLOP count is computed.

REFERENCES

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. *arXiv preprint arXiv:2108.07732*, 2021.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 7432–7439, 2020.

Lizhang Chen, Jonathan Li, Chen Liang, Ni Lao, and Qiang Liu. Training-free looped transformers. *arXiv preprint arXiv:2605.23872*, 2026.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Xinghao Chen, Anhao Zhao, Heming Xia, Xuan Lu, Hanlin Wang, Yanjun Chen, Wei Zhang, Jian Wang, Wenjie Li, and Xiaoyu Shen. Reasoning beyond language: A comprehensive survey on latent chain-of-thought reasoning. *arXiv preprint arXiv:2505.16782*, 2025.

Yung-Sung Chuang, Yujia Xie, Hongyin Luo, Yoon Kim, James R Glass, and Pengcheng He. Dola: Decoding by contrasting layers improves factuality in large language models. In *International Conference on Learning Representations*, volume 2024, pp. 54158–54183, 2024.

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. *arXiv preprint arXiv:1803.05457*, 2018.

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. *arXiv preprint arXiv:2110.14168*, 2021.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. Universal transformers. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HyzdRiR9Y7.

Hanwen Du, Yuxin Dong, and Xia Ning. Latent thinking optimization: Your latent reasoning language model secretly encodes reward signals in its latent thoughts. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 101000–101032, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/a4b1a25e1b37a17733c1fea50d0b7446-Paper-Conference.pdf.

Ying Fan, Yilun Du, Kannan Ramchandran, and Kangwook Lee. Looped Transformers for Length Generalization. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 14502–14520, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/25cc3adf8c85f7c70989cb8a97a691a7-Paper-Conference.pdf.

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

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. A framework for few-shot language model evaluation. Zenodo, 2023. lm-evaluation-harness.

Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach. *Advances in Neural Information Processing Systems*, 38: 41340–41391, 2025.

Angeliki Giannou, Shashank Rajput, Jy-Yong Sohn, Kangwook Lee, Jason D. Lee, and Dimitris Papailiopoulos. Looped Transformers as Programmable Computers. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 11398–11442. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/giannou23a.html.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason E Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=Itxz7S4Ip3.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, et al. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3828–3850, 2024.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=d7KBjmI3GmQ.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021. URL https://openreview.net/forum?id=qw8AKxfYbI.

HuggingFaceH4. AIME 2024. Hugging Face dataset, 2025. URL https://huggingface.co/datasets/HuggingFaceH4/aime_2024. 30 problems from the 2024 American Invitational Mathematics Examination, parts I and II.

Haoqiang Kang, Yizhe Zhang, Nikki Kuang, Nicklas Majamaki, Navdeep Jaitly, Yian Ma, and Lianhui Qin. Ladir: Latent diffusion enhances llms for text reasoning. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 81705–81733, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/843e1d8ab4819c8c8a4221fce2c8d455-Paper-Conference.pdf.

Tero Karras, Miika Aittala, Tuomas Kynkäänniemi, Jaakko Lehtinen, Timo Aila, and Samuli Laine. Guiding a diffusion model with a bad version of itself. *Advances in Neural Information Processing Systems*, 37:52996–53021, 2024.

Xiang Lisa Li, Ari Holtzman, Daniel Fried, Percy Liang, Jason Eisner, Tatsunori B Hashimoto, Luke Zettlemoyer, and Mike Lewis. Contrastive decoding: Open-ended text generation as optimization. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 12286–12312, 2023.

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 *Advances in Neural Information Processing Systems*, volume 36, pp. 21558–21572, 2023.

Weihao Liu, Dehai Min, and Lu Cheng. Latent thoughts tuning: Bridging context and reasoning with fused information in latent tokens. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=SFTFsixO2K.

Sean O'Brien and Mike Lewis. Contrastive decoding improves reasoning in large language models. *arXiv preprint arXiv:2309.09117*, 2023. URL https://arxiv.org/abs/2309.09117.

OpenCompass. AIME 2025. Hugging Face dataset, 2025. URL https://huggingface.co/datasets/opencompass/AIME2025. 30 problems from the 2025 American Invitational Mathematics Examination, parts I and II.

Hayden Prairie, Zachary Novack, Taylor Berg-Kirkpatrick, and Daniel Y Fu. Parcae: Scaling laws for stable looped language models. *arXiv preprint arXiv:2604.12946*, 2026.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 8732–8740, 2020. doi: 10.1609/aaai.v34i05.6399. URL https://ojs.aaai.org/index.php/AAAI/article/view/6399.

Guillaume Sanchez, Alexander Spangher, Honglu Fan, Elad Levi, and Stella Biderman. Stay on topic with classifier-free guidance. In *Forty-first International Conference on Machine Learning*, pp. 43197–43234, 2024. URL https://openreview.net/forum?id=RiM3cl9MdK.

Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J. Reddi. Reasoning with Latent Thoughts: On the Power of Looped Transformers. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 14855–14881, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/2676109d49d1eb26d6bc584a8f556305-Paper-Conference.pdf.

Weijia Shi, Xiaochuang Han, Mike Lewis, Yulia Tsvetkov, Luke Zettlemoyer, and Wen-tau Yih. Trusting your evidence: Hallucinate less with context-aware decoding. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pp. 783–791, 2024.

Lexiang Tang, Weihao Gao, Bingchen Zhao, Lu Ma, Qiao Jin, Bang Yang, and Yuexian Zou. Thinking by subtraction: Confidence-driven contrastive decoding for LLM reasoning. *arXiv preprint arXiv:2602.18232*, 2026. URL https://arxiv.org/abs/2602.18232.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. In *Advances in Neural Information Processing Systems*, volume 37, pp. 95266–95290, 2024.

Johannes Welbl, Nelson F Liu, and Matt Gardner. Crowdsourcing multiple choice science questions. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pp. 94–106, 2017.

Jonathan Williams, Olga Russakovsky, and Esin Tureci. Prioritize the Process, Not Just the Outcome: Rewarding Latent Thought Trajectories Improves Reasoning in Looped Language Models. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=cPAgJwcn3h.

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.

Jian Yang, Shawn Guo, Wei Zhang, Tianyu Zheng, Yaxin Du, Haau-Sing Li, Jiajun Wu, Yue Song, Yan Xing, Qingsong Cai, Zelong Huang, Chuan Hao, Ran Tao, Xianglong Liu, Wayne Xin Zhao, Mingjie Tang, Weifeng Lv, Ming Zhou, and Bryan Dai. LoopCoder-v2: Only Loop Once for Efficient Test-Time Computation Scaling. *arXiv preprint arXiv:2606.18023*, 2026. URL https://arxiv.org/abs/2606.18023.

Liu Yang, Kangwook Lee, Robert Nowak, and Dimitris Papailiopoulos. Looped Transformers are Better at Learning Learning Algorithms. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 42195–42214, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/b8402301e7f06bdc97a31bfaa653dc32-Paper-Conference.pdf.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 4791–4800, 2019.

Yuechi Zhou, Chuyue Zhou, Jianxin Zhang, Juntao Li, and Min Zhang. ALW: Adaptive Layer-Wise contrastive decoding enhancing reasoning ability in Large Language Models. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 8506–8524, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.447. URL https://aclanthology.org/2025.findings-acl.447/.

Rui-Jie Zhu, Zixuan Wang, Kai Hua, Tianyu Zhang, Ziniu Li, Haoran Que, Boyi Wei, Zixin Wen, Fan Yin, He Xing, Lu Li, Jiajun Shi, Kaijing Ma, Shanda Li, Taylor Kergan, Andrew Smith, Xingwei Qu, Mude Hui, Bohong Wu, Qiyang Min, Hongzhi Huang, Xun Zhou, Wei Ye, Jiaheng Liu, Jian Yang, Yunfeng Shi, Chenghua Lin, Enduo Zhao, Tianle Cai, Ge Zhang, Wenhao Huang, Yoshua Bengio, and Jason Eshraghian. Scaling latent reasoning via looped language models, 2026. URL https://arxiv.org/abs/2510.25741.

Ruike Zhu, Hanwen Zhang, Kevin Li, Tianyu Shi, Yiqun Duan, Chi Wang, Tianyi Zhou, Arindam Banerjee, and Zengyi Qin. Beyond Parameters: Exploring Virtual Logic Depth for Scaling Laws. *arXiv preprint arXiv:2506.18233*, 2025. URL https://arxiv.org/abs/2506.18233.

A LOOPED TRANSFORMERS AND LOOPCD

This appendix expands Sections 2 and 3: the state indexing and output interface behind Eq. (3), the four evaluated families, and the probability form of LoopCD-Logits.

A.1 STATES AND CODA LAYERS

LoopCD uses states at the end of complete iterations and the model's existing coda layers. The reference h_b and final state h_R enter the same output interface. LoopCD-Logits passes both states through this interface and combines the resulting logits. LoopCD-Hidden combines the states first and passes their result through it once. In each case, the modified logits feed the evaluation's existing scoring or token-selection rule.

Looped Transformers with a sandwich structure generally include their prediction head in the coda, whereas we count the coda separately from the final normalization and prediction head to better count the layers. Table A1 lists the resulting layer counts. The location of the shared block determines which coda layers must be repeated to obtain earlier logits.

Model	# Params	$\#\mathcal{P}$	$\#\mathcal{R}$	$\#\mathcal{C}$	R	L_{eff}
Ouro-1.4B	1.4B	0	24	0	4	96
Ouro-2.6B	2.6B	0	48	0	4	192
Huginn-0125 ($R = 32$)	3.5B	2	4	2	32	132
Huginn-0125 ($R = 16$)	3.5B	2	4	2	16	68
Parcae-370M	0.37B	4	4	4	8	40
Parcae-1.3B	1.3B	8	8	8	8	80
Looped-Qwen3	4.0B	15	4	17	8	64
Qwen3-4B	4.0B	36	0	0	–	36

Table A1: **The evaluated models share different portions of their computation.** $\#\mathcal{P}$, $\#\mathcal{R}$, and $\#\mathcal{C}$ count prelude, recurrent-block, and coda layers. The iteration count is R, and $L_{\text{eff}} = \#\mathcal{P} + R\#\mathcal{R} + \#\mathcal{C}$ counts layer applications. The Ouro-Thinking checkpoints used for reasoning share the listed Ouro architectures. Qwen3-4B is the dense checkpoint used by Looped-Qwen3. Layer counts alone do not give FLOPs because layers can differ in width and computation.

A.2 MODEL ARCHITECTURES

Ouro. Ouro repeats its decoder stack and trains predictions at intermediate depths (Zhu et al., 2026). The earlier state passes through the final normalization and vocabulary head to produce logits. There are no prelude or coda Transformer layers. A hidden-state update could be applied before the normalization and head, but our reported Ouro experiments use logits.

Huginn. Huginn computes a prelude representation, initializes a separate recurrent state with Gaussian noise, and conditions each recurrent update on the prelude representation (Geiping et al., 2025). The state denoted h_1 in this paper is the result of the first update, and h_0 is the noise initialization. Earlier logits require a pass through the coda and head. LoopCD-Hidden combines recurrent outputs before this pass.

Parcae. Parcae also initializes a separate recurrent state and repeatedly conditions on the normalized prelude representation. Its update includes learned linear terms together with the recurrent Transformer block (Prairie et al., 2026). The recurrent output is projected before entering the coda. LoopCD-Logits applies this projection, coda, and head to the earlier state. The hidden-state form combines states before the projection and uses the coda layers once.

Looped-Qwen3. The Training-Free Looped Transformers construction repeats a frozen window with damped residual updates. A substep has the form $u_{j+1} = (1 - 1/K)u_j + (1/K)g(u_j)$, where g is the frozen window and K is the number of substeps in the wrapper (Chen et al., 2026). For the evaluated model, the window contains four layers, with fifteen layers before it and seventeen after

it. Earlier logits require the frozen tail and head. The reduced-depth result in Figure 4 uses the first four of eight substeps, following the reported experiment. Changing the wrapper to four substeps with a different damping coefficient would define a different computation.

A.3 The probability form of LoopCD-Logits

Let Z_s and Z_w be the softmax normalizers of the final and earlier logits. Since $\log p_s(v) = z_s(v) - \log Z_s$ and $\log p_w(v) = z_w(v) - \log Z_w$, the quantity $(1+\omega)\log p_s(v) - \omega \log p_w(v)$ equals the guided logit $z'(v)$ of Eq. (3) plus a constant shared by all vocabulary entries. Exponentiating and normalizing therefore gives the probability form $p'(v) \propto p_s(v)\,(p_s(v)/p_w(v))^{\omega}$. The identity holds at each token position for its selected coefficient, including the adaptive rule. It describes the guided logits before any temperature or truncation used by an evaluation protocol. Nonlinear coda layers prevent applying this identity directly to LoopCD-Hidden.

B Detailed Experimental Setup

This appendix further details the evaluated model configurations, benchmark protocols, prompting strategies, and hyperparameter selection. Table A2 outlines the experimental coverage, while the subsequent subsections specify the evaluation procedures.

B.1 Model configurations and paired comparisons

Table A2 summarizes the evaluated combinations by model and task type. The standard iteration budget or the regular recurrent depth (R) is specified for each configuration (e.g., Huginn is evaluated at both $R = 16$ and $R = 32$). Ouro base and Thinking checkpoints are listed separately based on their task allocations. Looped-Qwen3 utilizes a frozen Qwen3-4B checkpoint with damped recurrent updates applied to a four-layer middle window (detailed in Appendix A.2 and Table A1).

To ensure rigorous paired comparisons, a LoopCD run and its unguided baseline always share the identical checkpoint, prompt, shot count, generation limit, stopping rule, answer extractor, and evaluator. Recurrent depths are also strictly matched, except in explicit ablation studies comparing standard and reduced budgets. For the reduced-budget study, standard-budget baselines average repeated logit runs where available, while hidden-state runs are paired with their own dedicated baselines.

		Full Recurrent Depth						Reduced Depth	
		Multiple choice		Greedy generation		Mathematical reasoning		Multiple choice	
Checkpoint	R	Logit	Hidden	Logit	Hidden	Logit	Hidden	Logit	Hidden
Ouro-1.4B / 2.6B (base)	4	✓	×	✓	×	×	×	×	×
Ouro-1.4B / 2.6B (Thinking)	4	×	×	×	×	✓	×	×	×
Huginn-0125	16, 32	✓	✓	✓	✓	×	×	✓	✓
Parcae-370M	8	✓	×	×	×	×	×	✓	×
Parcae-1.3B	8	✓	✓	×	×	×	×	✓	✓
Looped-Qwen3 (4B)	8	✓	✓	✓	✓	✓	×	×	✓

Table A2: **Experimental coverage by model and evaluation type.** ✓ indicates inclusion in the reported performance comparisons. × indicates exclusion. R gives the standard iteration budget. Logit and Hidden denote LoopCD-Logits and LoopCD-Hidden. Reduced iteration budget experiments use multiple-choice benchmarks only (Huginn reduced from 32 to 16, Parcae and Looped-Qwen3 from 8 to 4).

B.2 BENCHMARKS AND TABLE CONVENTIONS

All reported scores are percentages. A subscript on a guided score denotes its signed percentage-point change (Δ) from the strictly matched unguided baseline. The "Mean Δ" aggregates these changes with equal weight across the specific columns relevant to each evaluation suite.

Multiple-choice benchmarks. Evaluated on ARC-Challenge, ARC-Easy (Clark et al., 2018), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2020), PIQA (Bisk et al., 2020), SciQ (Welbl et al., 2017), and MMLU (Hendrycks et al., 2021). Candidates are scored by likelihood using the language model evaluation harness (Gao et al., 2023).

Greedy generation. Evaluated on HumanEval (Chen et al., 2021), MBPP (Austin et al., 2021), GSM8K (Cobbe et al., 2021), and MMLU-Pro (Wang et al., 2024) with greedy decoding. Code generation is assessed via EvalPlus (Liu et al., 2023) using both base and extended tests. GSM8K-CoT uses a strict-match extractor for the `#### N` format.

Mathematical reasoning. Evaluated on AIME 2024 (HuggingFaceH4, 2025), AIME 2025 (OpenCompass, 2025) (30 problems each, integer matching), and the 674-problem English mathematics subset of OlympiadBench (He et al., 2024) (symbolic equivalence). Each AIME set combines parts I and II of that year's examination. We estimate pass@1 and pass@10 separately from 16 sampled solutions per problem using the standard unbiased estimator (Chen et al., 2021; Zhu et al., 2026).

B.3 PROMPTS AND SAMPLING

Tables A3 and A4 document the few-shot configurations for multiple-choice scoring and generation. Code benchmarks operate zero-shot while retaining the standard EvalPlus chat instruction. MMLU-Pro uses five chain-of-thought (CoT) demonstrations. GSM8K is evaluated with three demonstrations by default, except for GSM8K-CoT (which uses eight) and all hidden-state generation runs. Mathematical reasoning strictly uses zero-shot sampling.

Model	ARC-C	ARC-E	HellaSwag	WinoGrande	PIQA	SciQ	MMLU
LoopCD-Logits							
Ouro-1.4B / 2.6B	25	8	10	5	0	0	5
Huginn-0125 R=32 / R=16	0	0	0	0	0	0	0
Parcae-370M / 1.3B	25	0	0	0	0	0	5
Looped-Qwen3	25	0	0	0	0	0	5
LoopCD-Hidden							
Huginn-0125	0	0	0	0	0	0	0
Parcae-1.3B	25	0	0	0	0	0	5
Looped-Qwen3	0	0	0	0	0	0	0

Table A3: **Multiple-choice shot counts remain fixed within each paired comparison.** Differences across model families and logit/hidden-state settings limit direct comparisons of absolute scores.

Model	GSM8K	MMLU-Pro	HumanEval(+)	MBPP(+)
LoopCD-Logits				
Ouro-1.4B / 2.6B	3	5	0	0
Huginn-0125 R=32 / R=16	8†	5	0	0
Looped-Qwen3	3	5	0	0
LoopCD-Hidden				
Huginn-0125 R=32 / R=16	–	–	0	0

Table A4: **Prompting configurations across generation benchmarks.** † denotes GSM8K-CoT. Code benchmarks use no worked examples. HumanEval(+) and MBPP(+) include both base and extended tests.

B.4 STRENGTHS AND REFERENCE STATES

Hyperparameters are determined via screening sweeps prior to full-suite evaluations (see Appendix G.1 and Figures A11 and A12). Table A5 records the selected values for experiments at standard recurrent depth.

Configuration	Fixed ω	Cap $\omega_{\max}$
Multiple-choice scoring, LoopCD-Logits		
Ouro-1.4B	0.5	1.0
Ouro-2.6B	0.5	1.0
Huginn-0125 R=32	0.5	0.5
Huginn-0125 R=16	0.5	0.5
Parcae-370M	0.5	1.0
Parcae-1.3B	0.5	1.0
Looped-Qwen3	0.5	1.0
Multiple-choice scoring, LoopCD-Hidden		
Huginn-0125 R=32	0.5, $h_b = h_6$	–
Huginn-0125 R=16	0.5, $h_b = h_7$	–
Parcae-1.3B R=8	1.0, $h_b = h_1$	–
Generation, LoopCD-Logits		
Ouro-1.4B	0.3	0.5
Ouro-2.6B	0.3	1.0
Huginn-0125 R=32	0.2	0.25
Huginn-0125 R=16	0.2	0.25
Looped-Qwen3	0.3	0.5
Generation, LoopCD-Hidden		
Huginn-0125 R=32	0.3, $h_b = h_7$	–
Huginn-0125 R=16	0.3, $h_b = h_6$	–
Mathematical reasoning, LoopCD-Logits		
Ouro-1.4B-Thinking	0.5	1.0
Ouro-2.6B-Thinking	0.5	1.5
Looped-Qwen3	0.4	0.5

Table A5: **Selected hyperparameters for standard-budget experiments.** ω is fixed strength, and $\omega_{\max}$ is the adaptive cap. LoopCD-Hidden rows specify the reference state h_b to avoid initialization noise. LoopCD-Logits rows use h_1. Dashes denote untested modes. Reduced-depth configurations follow Table A9.

Regarding strength-allocation strategies, we evaluate both fixed and adaptive strengths for regular-depth logit-space setups (multiple-choice scoring, greedy generation, and reasoning on Ouro-Thinking and Looped-Qwen3). Conversely, fixed strength alone is applied to all hidden-state configurations and to all experiments operating under a reduced recurrent depth.

For LoopCD-Logits, the fixed strength (ω) is 0.5 for multiple-choice scoring. For answer and code generation, it is 0.3 for Ouro and Looped-Qwen3 and 0.2 for Huginn. Reasoning strengths and adaptive caps ($\omega_{\max}$) are listed in Table A5.

For LoopCD-Hidden, the reference state (h_b) is chosen with the model's initialization in mind (Appendix E.1). Parcae-1.3B has a deterministic prelude, so its contrast uses h_1 at $R = 8$ with $\omega = 1.0$. Huginn starts from Gaussian noise. After its burn-in phase, multiple-choice scoring uses h_6 at $R = 32$ and h_7 at $R = 16$, both with $\omega = 0.5$. Generation uses h_7 and h_6 at those respective depths, with $\omega = 0.3$. Reduced-depth experiments use the distinct settings in Table A9.

C EXTENDED RESULTS IN FULL RECURRENT DEPTH

This appendix supplements Sections 4.1 and 4.2 with GSM8K and MMLU-Pro results under LoopCD-Logits and a comparison of Looped-Qwen3's reasoning metrics. The LoopCD-Hidden reference states and strengths are specified in Appendix B.4.

C.1 GSM8K AND MMLU-PRO UNDER LOOPCD-LOGITS

Unlike the likelihood-based or code-execution evaluations, GSM8K and MMLU-Pro are scored based on the final generated answer. We evaluate GSM8K using a three-shot setup (eight-shot with a chain of thought for Huginn). For MMLU-Pro, which covers MMLU's subject matter, we use five-shot chain-of-thought prompting, contrasting with Table 2 where MMLU is scored by likelihood.

Table A6 presents the five generation configurations under LoopCD-Logits. MMLU-Pro improves in nine out of the ten rows, gaining up to $+1.42$ points on Ouro-2.6B under the fixed rule. In contrast, GSM8K exhibits mixed results: it improves in six of the ten rows (by up to $+2.04$ points on Ouro-1.4B), but declines by up to -1.29 on Huginn-0125 at $R = 32$. Note that the two Parcae models

are excluded from these benchmarks: Parcae-1.3B performs at the random floor on GSM8K, and MMLU-Pro's prompts exceed Parcae's context window limit.

	GSM8K			MMLU-Pro		
Model	Baseline	LoopCD, fixed	LoopCD, adaptive	Baseline	LoopCD, fixed	LoopCD, adaptive
Ouro-1.4B	75.82	$77.48_{+1.66}$	$77.86_{+2.04}$	48.87	$49.14_{+0.27}$	$48.88_{+0.01}$
Ouro-2.6B	81.27	$82.34_{+1.07}$	$80.44_{-0.83}$	56.08	$57.50_{+1.42}$	$56.99_{+0.91}$
Huginn-0125 R=32	42.91	$41.62_{-1.29}$	$43.06_{+0.15}$	13.59	$14.50_{+0.91}$	$14.23_{+0.64}$
Huginn-0125 R=16	38.21	$39.80_{+1.59}$	$38.44_{+0.23}$	13.66	$14.26_{+0.60}$	$14.18_{+0.52}$
Looped-Qwen3	85.29	$84.15_{-1.14}$	$84.23_{-1.06}$	58.86	$58.85_{-0.01}$	$59.32_{+0.46}$

Table A6: **MMLU-Pro improves in nine of ten rows, while GSM8K moves in both directions.** The five generation configurations are evaluated at their full recurrent depth under LoopCD-Logits. Baselines are shaded in gray. Subscripts give the signed percentage-point change from the matched baseline (Appendix B.2).

C.2 Reasoning metrics: pass@1 and pass@10 across sampled trajectories

The reasoning gains observed in Table 1 are an order of magnitude larger than the changes on the multiple-choice suite in Table 2. The evaluation protocol offers a plausible explanation: a multiple-choice question alters only a single likelihood comparison, whereas a sampled reasoning solution involves hundreds of sequential token decisions, each of which guidance can shift. Furthermore, pass@k rewards finding a correct answer among the k generated samples.

This compounding helps explain the divergence in Looped-Qwen3's mathematical-reasoning metrics (Table A7), the only block in the reasoning suite where pass@1 declines while pass@10 rises. On AIME 2024, fixed-strength guidance lowers pass@1 from 64.79 to 61.88: a single sampled solution succeeds less often. Yet pass@10 rises from 82.91 to 86.17, a three-point increase on a thirty-problem set, or roughly one additional problem solved at least once among ten samples. This pattern is consistent with guidance broadening coverage across sampled trajectories even as it reduces per-sample success on some problems. The aggregate pass@k scores do not directly measure trajectory diversity. For the Ouro-Thinking models, both metrics rise across nearly every configuration.

	AIME 2024		AIME 2025		OlympiadBench		Mean Δ	
Model	pass@1	pass@10	pass@1	pass@10	pass@1	pass@10	pass@1	pass@10
Looped-Qwen3	64.79	82.91	55.00	75.68	66.52	78.91	–	–
LoopCD, fixed	$61.88_{-2.91}$	$86.17_{+3.26}$	$54.58_{-0.42}$	$77.42_{+1.74}$	$66.52_{+0.00}$	$79.60_{+0.69}$	−1.11	+1.90
LoopCD, adaptive	$61.04_{-3.75}$	$86.10_{+3.19}$	$53.96_{-1.04}$	$81.28_{+5.60}$	$66.29_{-0.23}$	$79.65_{+0.74}$	−1.67	+3.18

Table A7: **Looped-Qwen3 on mathematical reasoning: pass@1 declines while pass@10 rises.** Both metrics are estimated separately from sixteen sampled solutions per problem, following Appendix B.2. The divergence is consistent with broader coverage across samples under guidance.

D Computation and Cost Analysis

This appendix expands upon Section 4.3, detailing the general FLOP accounting for the two guidance forms, followed by a comprehensive evaluation of the reduced-depth settings, covering both their performance recovery and computational tradeoffs. Finally, we examine the underlying representational dynamics that differentiate LoopCD-Logits and LoopCD-Hidden.

D.1 FLOP accounting

We calculate theoretical computational cost by assuming two FLOPs per multiply-and-accumulate operation in matrix products during a 512-token prefill. The unguided baseline reflects the standard output computation. LoopCD-Logits introduces model-specific overheads to process the reference

state. Specifically, Ouro requires normalization and the language modeling head. Huginn adds the coda and head. Parcae utilizes the projection, coda, and head. Finally, Looped-Qwen3 applies the frozen tail and head. Table A8 summarizes the resulting computational multipliers and their architectural breakdown.

		LoopCD-Logits				
Model	Forward TFLOPs	FLOPs ×	head (%)	coda or tail (%)	Layers after the loop	LoopCD-Hidden ×
Ouro-1.4B	5.360	1.019	1.92	0.00	0	1.000
Ouro-2.6B	10.617	1.010	0.97	0.00	0	1.000
Huginn-0125 R=32	54.526	1.022	0.65	1.51	2	1.000
Huginn-0125 R=16	28.261	1.042	1.25	2.90	2	1.000
Parcae-370M	0.593	1.152	5.80	9.42	4	1.000
Parcae-1.3B	2.500	1.119	2.06	9.79	8	1.000
Looped-Qwen3	7.287	1.306	5.47	25.11	17	1.000

Table A8: **LoopCD-Logits adds computational overhead based on post-loop layers, whereas LoopCD-Hidden adds negligible cost.** Forward TFLOPs represent one unguided 512-token prefill. The logit multiplier is relative to this baseline, with the extra work divided between the head and the coda/frozen tail applied to the reference state. LoopCD-Hidden relies solely on vector operations before a single output pass, yielding a $1.000\times$ multiplier.

Model	Guidance	Full → reduced	Reference	Fixed ω
Huginn-0125	Logit	$32 \to 16$	h_5	0.5
Huginn-0125	Hidden-state	$32 \to 16$	h_6	0.5
Parcae-370M	Logit	$8 \to 4$	h_1	0.75
Parcae-1.3B	Logit	$8 \to 4$	h_1	0.5
Parcae-1.3B	Hidden-state	$8 \to 4$	h_1	0.75
Looped-Qwen3	Hidden-state	$8 \to 4$	h_1	1.0

Table A9: **Settings for the six reduced-depth comparisons in Figure 4.** Each guided model uses half as many recurrent iterations as its full-depth baseline. Looped-Qwen3 exits after the first four of eight damped substeps, retaining the coefficient $1/8$ used at full depth. References use the paper's indexing: h_1 follows the first iteration. Strength is fixed uniformly across benchmarks within each setting.

Conversely, LoopCD-Hidden only introduces lightweight vector operations and relies on a single standard coda pass. Consequently, its computational multiplier effectively rounds to $1.000\times$ in this accounting, representing negligible overhead.

Note that these FLOP counts reflect theoretical arithmetic workload rather than end-to-end wall-clock speedups. Additionally, the specific reduced-depth experiments discussed subsequently are evaluated exclusively on the multiple-choice suite.

D.2 Reduced-depth evaluation: performance recovery and computational tradeoffs

Having established the baseline computational overheads, we now evaluate configurations where the recurrent iteration budget is strictly halved. Table A9 details the specific operational parameters for the experiments presented in Figure 4. To ensure a rigorous evaluation, the fixed strength for each model is determined uniformly across the multiple-choice suite and held constant for all benchmarks. Because halving the iteration budget inherently alters the models' recurrent dynamics, these operational parameters can differ from their full-depth counterparts. For instance, to accommodate the shortened trajectory, LoopCD-Logits on Huginn utilizes a later reference state (h_5) compared to the initial state (h_1) used at full depth.

As reported in Section 4.3, halving the recurrent iterations degrades the unguided baseline, but LoopCD successfully recovers this deficit across all evaluated configurations.

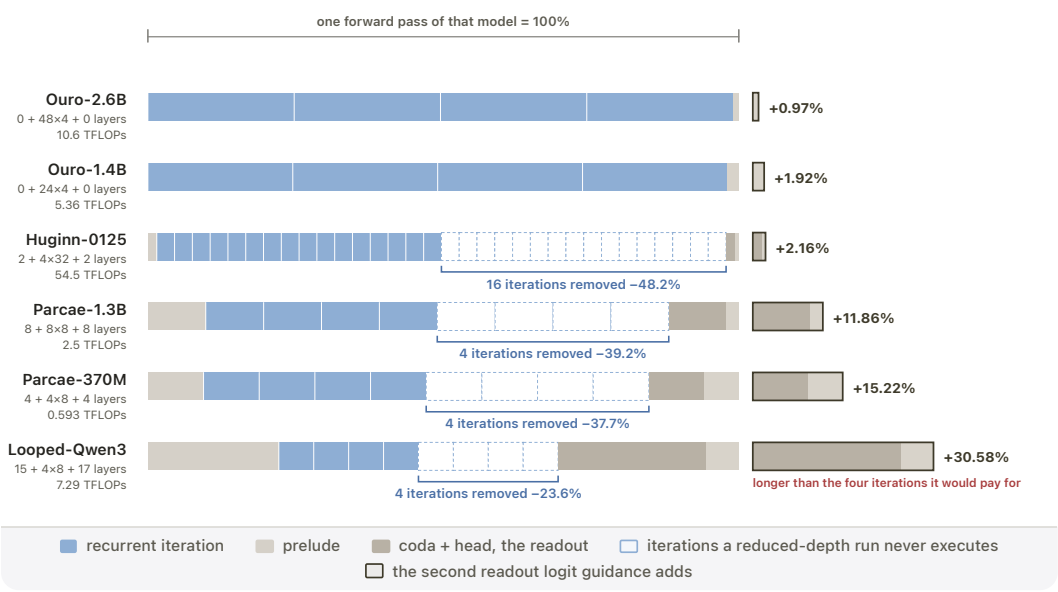

Figure A1: **Computational breakdown of the forward pass and guidance overhead.** Each bar scales the forward pass of a given configuration, normalized to its total compute. The pass includes the prelude, recurrent iterations, followed by the coda and head used for final distribution extraction. Dashed blocks represent the skipped iterations in a reduced-depth run, highlighting the computational savings. The outlined block on the right represents the overhead of the second coda and head evaluation required by LoopCD-Logits. Architectures with a larger recurrent proportion save more compute and incur less overhead. For Looped-Qwen3, the readout overhead exceeds the savings from skipped iterations. LoopCD-Hidden introduces negligible overhead and thus adds no visible blocks. All theoretical FLOP counts assume a 512-token prefill.

To support the computational tradeoff analysis in the main text, Figure A1 visualizes the structural breakdown of each model's forward pass. Because depth reduction only removes recurrent blocks, the potential savings are strictly bounded by the loop's relative share of the total computation. Conversely, the LoopCD-Logits overhead is dictated entirely by the non-recurrent layers following the loop (the coda and prediction head). Figure A1 physically illustrates why recurrence-dominated architectures like Huginn benefit immensely from LoopCD-Logits at reduced depths, whereas retrofitted architectures like Looped-Qwen3, which possess massive non-recurrent components, experience readout overheads that exceed their loop savings, necessitating LoopCD-Hidden for effective FLOP reduction.

Setting	Iterations	Forward FLOPs at half depth, guidance included (fraction of the full pass)	Removed (%)
Huginn-0125, logits	$32 \rightarrow 16$	0.54	46.0%
Huginn-0125, hidden states	$32 \rightarrow 16$	0.52	48.2%
Parcae-370M, logits	$8 \rightarrow 4$	0.78	22.5%
Parcae-1.3B, logits	$8 \rightarrow 4$	0.73	27.3%
Parcae-1.3B, hidden states	$8 \rightarrow 4$	0.61	39.2%
Looped-Qwen3, hidden states	$8 \rightarrow 4$	0.76	23.6%

Table A10: **Each reduced-depth setting spends about half to three quarters of the full-depth forward pass, guidance included.** Analytic forward FLOPs at a 512-token prefill are expressed as a fraction of the same checkpoint's unguided full-depth pass. The logit settings strictly include their second readout overhead.

Table A10 provides the exact computational expenditures for all six reduced-depth settings analyzed in Figure 4, detailing the specific fractional FLOP costs that define the shifted Pareto frontier.

D.3 Representational dynamics: LoopCD-Logits versus LoopCD-Hidden

While the preceding subsections establish the computational tradeoffs and performance recovery of the two guidance forms, they also differ fundamentally in their representational dynamics. Specifically, they differ in how effectively performance gains are retained across different model architectures, and in how sensitive the final output distribution is to the guidance strength.

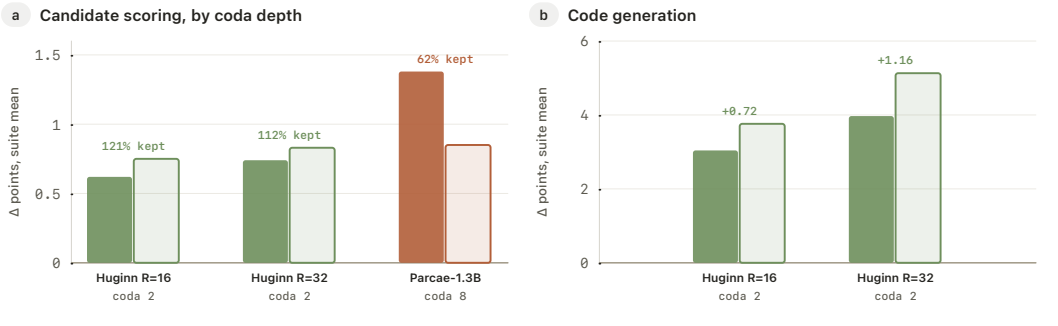

Figure A2: **LoopCD-Hidden preserves logit-space gains behind shallow codas and surpasses them in code generation. (a)** Multiple-choice: Seven-benchmark mean score changes at the selected settings from Tables 2 and 3. LoopCD-Logits (solid bars) is compared against LoopCD-Hidden (outlined bars) across three configurations, ordered by increasing coda depth. The annotated percentages indicate the proportion of the logit gain retained by the hidden-state form. **(b)** Code generation: Four-column mean score changes at the same settings for Huginn at both depths, highlighting the absolute difference between the two forms. Each guidance form is evaluated strictly against its respective unguided baseline.

Impact of coda depth on hidden-state retention. Comparing Table 3 to Table 2 reveals that LoopCD-Hidden retains less of the logit gain as the architecture's coda deepens (Figure A2(a)). For Huginn, which has a shallow two-layer coda, LoopCD-Hidden achieves nearly identical gains to fixed LoopCD-Logits on the seven-benchmark mean. Specifically, it yields 0.83 versus 0.74 points at $R = 32$, and 0.75 versus 0.62 points at $R = 16$. Conversely, on Parcae-1.3B, which features a deeper eight-layer coda, LoopCD-Hidden retains only roughly sixty percent of the logit gain (0.85 versus 1.38).

Ouro presents the opposite extreme, as it possesses zero layers after the recurrent loop. Consequently, on HellaSwag, LoopCD-Hidden outperforms LoopCD-Logits for both Ouro-1.4B (+1.73 versus +0.76) and Ouro-2.6B (+2.61 versus +1.56).

This behavior can be explained by representation attenuation. A guidance contrast injected before the coda is progressively damped by every subsequent layer. Note that while this structural trend holds broadly, the reported hidden-state rows optimize their own references and strengths independently (Appendix B.4). Within multiple-choice benchmarks specifically, the logit form maintains its advantage primarily on ARC-Challenge and HellaSwag for all three architectures with a coda. On most other benchmarks, the hidden form matches or exceeds logit performance, with Parcae-1.3B being the notable exception (Figure A3).

Performance retention in generative tasks. Extending this analysis to generative benchmarks, the hidden-state form actually exceeds the fixed logit form across both evaluated Huginn depths (Figure A2(b)). Evaluated against their respective baselines in code generation tasks, LoopCD-Hidden achieves 5.13 versus 3.97 points on the four-column mean at $R = 32$, and 3.76 versus 3.04 points at $R = 16$.

Output sensitivity to guidance strength. When analyzing the distributional updates directly, identical strength values (ω) yield different magnitudes of output divergence depending on the architecture. Specifically, applying guidance in the hidden-state space produces a larger output divergence

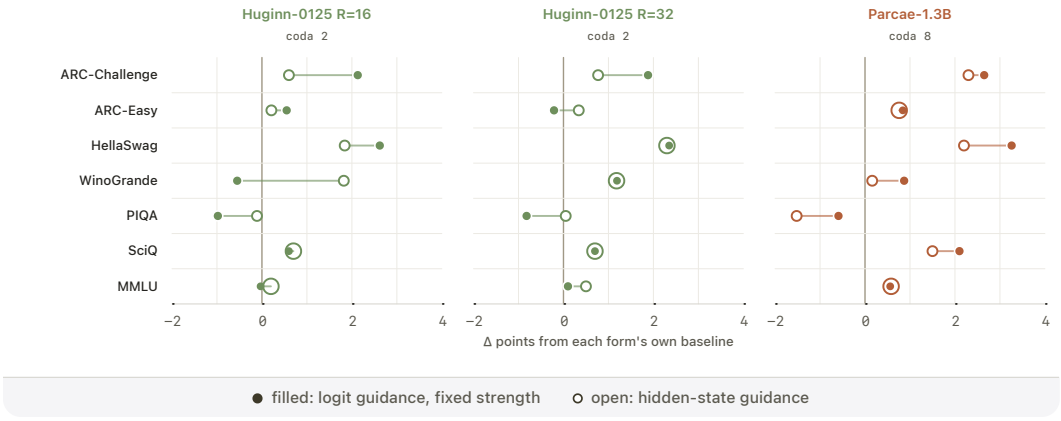

Figure A3: **Distribution of the logit form's performance advantage across multiple-choice benchmarks.** Absolute score changes for each benchmark at the selected settings from Tables 2 and 3. LoopCD-Logits (filled markers) is compared against LoopCD-Hidden (open markers), with both forms measured relative to their own unguided baselines. Benchmarks are displayed across rows, while configurations possessing a coda are arranged in columns ordered by increasing coda depth.

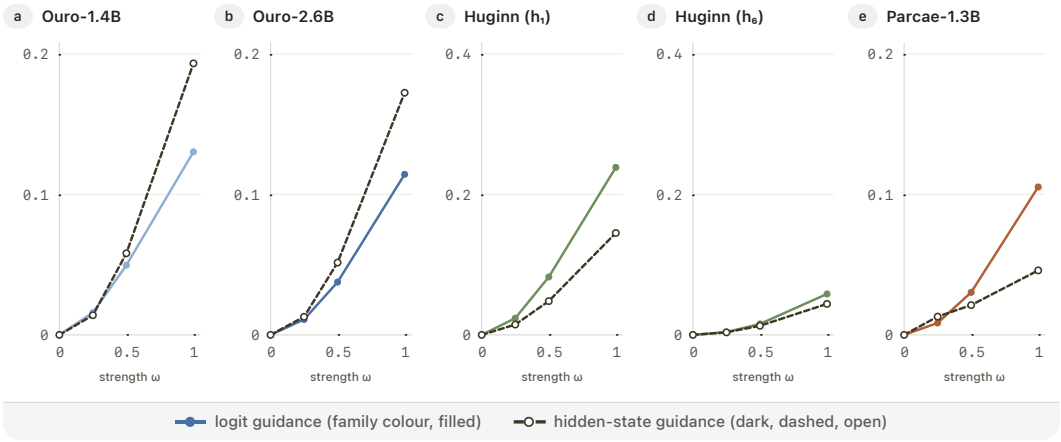

Figure A4: **Output sensitivity to guidance strength depends on coda depth.** Mean per-position Jensen–Shannon divergence (in bits) between the guided and final unguided distributions across varying strengths on 300 HellaSwag documents. LoopCD-Logits is shown in the family color, LoopCD-Hidden in dark gray. **(a-b)** Ouro variants (no coda) exhibit heightened sensitivity to LoopCD-Hidden. **(c-d)** Huginn variants at $R = 32$, using h_1 and h_6 respectively. **(e)** Parcae-1.3B (deep coda) shows damped hidden-state sensitivity. Ouro panels share a y-axis scale, as do Huginn panels. Parcae's `bf16` non-repeatability imposes a 0.006-bit empirical floor.

for models whose entire stack loops, but a substantially smaller divergence for models with a deep coda (Figure A4, Table A11).

		$\omega = 0.5$		$\omega = 1$	
Model	Reference	hidden states	logits	hidden states	logits
Ouro-1.4B	h_1	0.058	0.050	0.19	0.13
Ouro-2.6B	h_1	0.052	0.038	0.17	0.11
Huginn-0125 R=32	h_1	0.048	0.083	0.15	0.24
Huginn-0125 R=32	h_6	0.013	0.016	0.044	0.058
Parcae-1.3B	h_1	0.021	0.030	0.046	0.105

Table A11: **Output divergence at fixed guidance strengths varies by architecture.** The table reports the mean per-position Jensen–Shannon divergence (in bits) between the guided distribution and the final unguided prediction over 300 HellaSwag documents. LoopCD-Hidden induces a larger step in Ouro (no coda) but a smaller step in models with a coda. Due to non-deterministic `bf16` operations, Parcae's forward pass carries an empirical divergence floor of 0.006 bits between two identical passes.

In Ouro, only layer normalization separates the recurrent state from the vocabulary head. Consequently, LoopCD-Hidden shifts the output distribution much more aggressively than LoopCD-Logits at equivalent strengths. Because of this heightened sensitivity, Ouro's optimal hidden-state gain peaks at lower strength values compared to its logit gain (0.25 versus 0.5 for Ouro-1.4B, and 0.5 versus 1 for Ouro-2.6B).

ARC-Challenge

Reference k	Alone (%)	Disagrees (%)	Gain at $\omega = 0.5$
Huginn-0125, final prediction 37.54%			
1	22.78	51.3	3.07
2	22.10	46.8	1.79
4	30.12	35.8	0.94
6	32.25	29.5	1.11
8	33.45	20.0	0.60
16	37.20	7.3	0.17
24	38.05	2.7	0.17
Parcae-1.3B, final prediction 40.36%			
1	32.34	30.0	3.50
2	38.91	18.1	2.39
4	40.36	9.7	1.54
6	41.04	7.0	1.02
Ouro-1.4B, final prediction 60.41%			
1	38.65	40.2	2.13
2	54.86	17.0	0.09
3	59.56	6.1	0.34
Ouro-2.6B, final prediction 66.21%			
1	47.95	30.6	2.56
2	62.29	10.7	0.51
3	65.36	3.4	0.60

HellaSwag

Reference k	Alone (%)	Disagrees (%)	Gain at $\omega = 0.5$
Huginn-0125, final prediction 66.74%			
2	34.40	44.2	4.31
4	46.35	27.6	4.12
6	56.36	15.3	2.58
8	61.30	9.4	1.47
16	66.33	2.3	0.09
24	66.62	0.8	0.05
Parcae-1.3B, final prediction 56.28%			
1	44.39	19.9	2.90
2	50.76	11.4	1.68
4	55.01	5.7	0.47
6	56.09	3.6	0.13
Ouro-1.4B, final prediction 74.52%			
1	55.39	31.1	0.76
2	71.47	10.8	0.33
3	73.95	3.1	0.03
Ouro-2.6B, final prediction 79.47%			
1	68.89	18.0	1.56
2	77.84	6.6	0.86
3	79.23	2.6	0.03

Table A12: **Guidance gain depends on disagreement rather than standalone accuracy.** LoopCD-Logits evaluated with reference iteration k at $\omega = 0.5$. ARC-Challenge uses 25-shot prompting for Ouro and Parcae, and zero-shot for Huginn at $R = 32$. HellaSwag evaluates 10-shot for Ouro and zero-shot for all others. Each block's header reports the final prediction's baseline accuracy. "Alone" denotes the accuracy of the reference when read out independently.

Conversely, a coda damps the hidden-state update. For Parcae (eight coda layers) and Huginn, LoopCD-Logits moves the distribution further than LoopCD-Hidden at identical strengths, entirely reversing the trend seen in Ouro. Interestingly, Huginn's hidden-state form utilizing the first state (h_1) still peaks early (0.15 versus 0.25). This exception occurs because the initial-state update is only

moderately aligned with the true re-ranking direction and instead carries a large noise component orthogonal to the final logits (Figure A10), as analyzed in Appendix E.1.

E THE REFERENCE ITERATION

This appendix expands upon Section 5.1 by deeply analyzing the dynamics of intermediate iterations. We first examine how standalone accuracy and disagreement drive reference selection, including the specific effects of noise initialization. We then map the geometric convergence of these states in representation space, and finally analyze their intra-iteration dynamics in the output space to explain convergence behavior.

E.1 REFERENCE SELECTION ACROSS ITERATIONS

To understand what makes an effective reference, Table A12 details the standalone accuracy, disagreement with the final prediction, and resulting guidance gain for every evaluated reference iteration k. The empirical results demonstrate that guidance gain scales reliably with the reference's disagreement from the final prediction, rather than its standalone competence. Figure A5 shows how standalone accuracy climbs across the iterations.

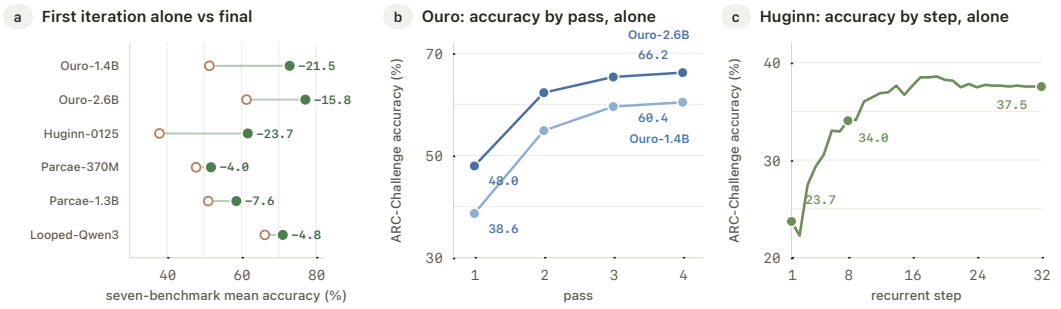

Figure A5: **The first iteration's prediction is much weaker than the final one, and accuracy climbs across the iterations. (a)** After the first iteration (open) against the final one (filled), seven-benchmark mean accuracy, six models. **(b)** Ouro after each pass, ARC-Challenge, 25-shot. **(c)** Huginn after each of its 32 steps, ARC-Challenge, zero-shot.

For instance, Parcae-1.3B's sixth step achieves a highly competent standalone accuracy of 41.04% compared to the final prediction's 40.36%. However, because it has largely converged with the final state, it yields only a single point of gain. Conversely, Huginn's second step is just as weak as its first (22.10% against 22.78%), yet because its disagreement has already shrunk, it provides only half the guidance gain of the initial step. Consequently, the first iteration universally serves as the optimal logit reference across stable architectures.

This reliance on divergence direction also explains a notable architectural anomaly: Huginn's hidden-state reference performance. As shown in Figure 6(b), Huginn's hidden-state sweep is negative at the first state and peaks later at the sixth. This occurs because Huginn initializes its recurrent state from Gaussian noise and subsequently moves it on a hypersphere. Across its 32 steps, the state's norm stays tightly bounded at 76.4 within a spread of 0.05. The first step is nearly as long as the radius itself, making the whole path 6.0 times longer than the net displacement. The trajectory behaves like a random walk for every one of the 1,172 ARC-Challenge questions.

Therefore, the contrast between the first hidden state and the final one is heavily dominated by the initial noise-removal step, not by a meaningful change in the prediction itself. Because the coda layers and head project this initial noise away, the first logits form an excellent reference. However, LoopCD-Hidden directly amplifies this unprojected noise, rendering the first hidden state detrimental.

E.2 GEOMETRIC CONVERGENCE IN REPRESENTATION SPACE

Transitioning from output accuracy to internal representations, Figures A6 and A7 visualize how intermediate states converge geometrically toward the final state. To isolate meaningful structural changes, every hidden dimension is standardized using its mean and variance pooled over the scored positions and unrolled layers. Let $\tilde{h}_W$ and $\tilde{h}_S$ denote the standardized intermediate state and the standardized final state, respectively. The geometric convergence is measured using cosine similarity:

$$\text{sim}(\tilde{h}_W, \tilde{h}_S) = \frac{\tilde{h}_W \cdot \tilde{h}_S}{\|\tilde{h}_W\|_2 \|\tilde{h}_S\|_2} \tag{5}$$

Computing the similarity strictly after this standardization drastically reduces the distortion caused by dimensions with excessively large variances.

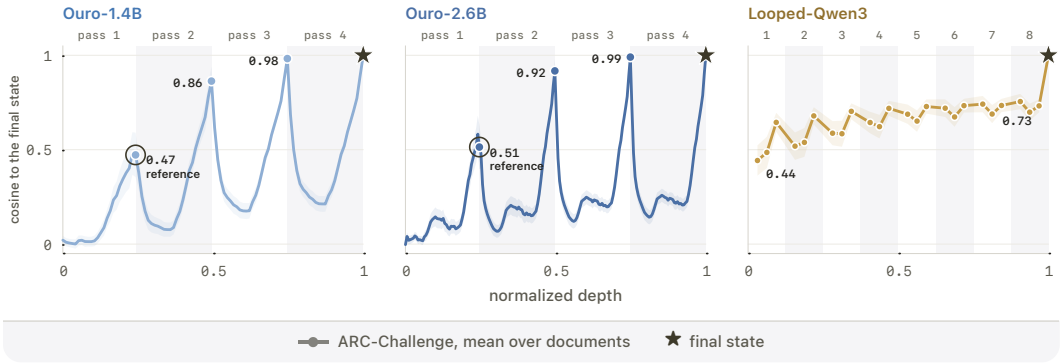

Figure A6: **Trajectory of hidden states toward the final representation.** Standardized cosine similarity between intermediate states and the final state across unrolled computation layers on ARC-Challenge. Alternating bands delineate Ouro's passes and Looped-Qwen3's substeps. Dots indicate the states at the end of each Ouro pass (values annotated). The ringed dot identifies the reference state utilized in Tables 1 to 2. Shaded error bands represent the variance across evaluated documents. The star indicates the final state.

These visualizations highlight distinct architectural dynamics. As shown in Figure A6, Ouro exhibits a sharp geometric reset. Its similarity to the final state drops dramatically after every pass boundary, followed by a steady recovery within the pass itself. In contrast, Looped-Qwen3 incorporates damped substeps that constrain the updates, resulting in a much smoother, gradual climb toward the final representation.

E.3 INTRA-ITERATION DYNAMICS AND OUTPUT CONVERGENCE

While the geometric view maps hidden-state trajectories, analyzing the output space explains why internal layers fail as references and how models ultimately settle.

Figure A8(a) measures the output divergence between each unrolled intermediate prediction and the final prediction. This difference is quantified by the Jensen–Shannon divergence (JSD), a smoothed and symmetric version of the Kullback–Leibler (KL) divergence. For an intermediate output distribution P and the final distribution Q, the JSD is defined as:

$$\text{JSD}(P \parallel Q) = \frac{1}{2} D_{\text{KL}}(P \parallel M) + \frac{1}{2} D_{\text{KL}}(Q \parallel M) \tag{6}$$

where $M = \frac{1}{2}(P + Q)$ is the mixture distribution.

Crucially, inside every pass, the prediction distribution P drifts wildly. For Ouro-1.4B, the JSD at the end of the first three passes sits tightly at 0.71, 0.13, and 0.013 bits (0.56, 0.06, and 0.008 for Ouro-2.6B). However, across the internal layers within those passes, the divergence spikes to an average of 0.57 to 0.84 bits, remaining just as erratic in the final pass as in the first.

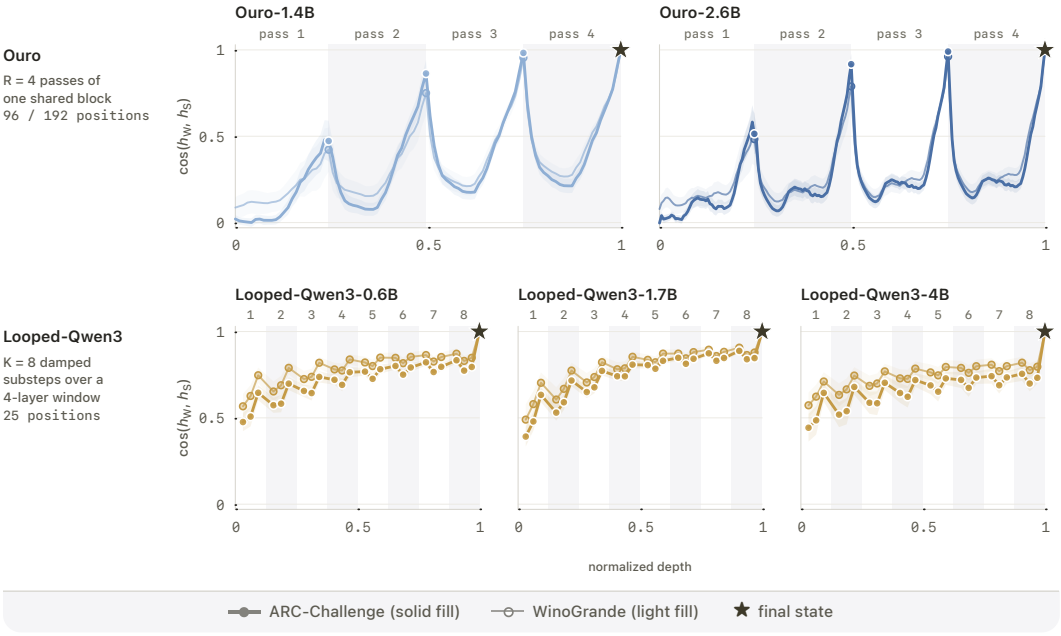

Figure A7: **Standardized representation similarity across varying recurrent architectures.** Cosine similarity to the final state (S) is plotted over normalized computational depth for intermediate states (W) across ten configurations. Ouro demonstrates pronounced drops strictly at iteration boundaries, whereas Looped-Qwen3's damped substeps yield smoother similarity progressions. Bands indicate the spread across documents. The star marks the final state's self-similarity limit.

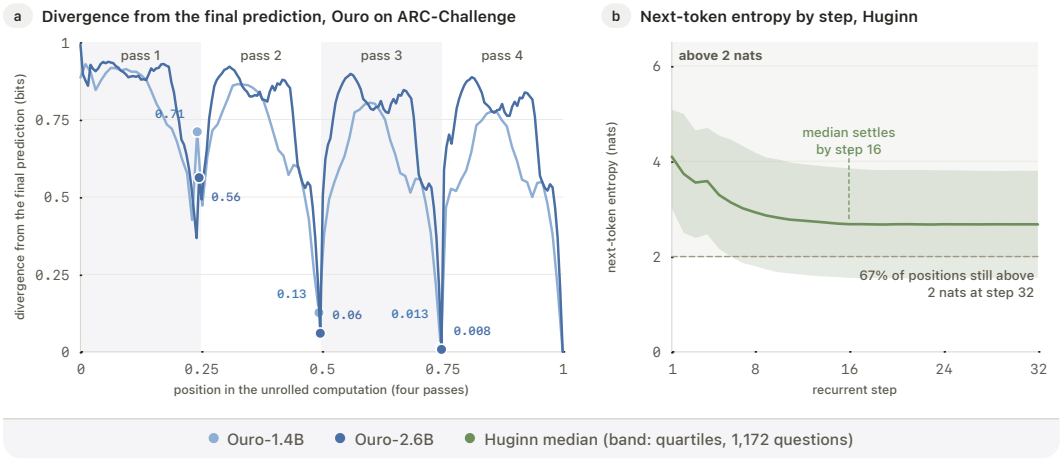

Figure A8: **Intra-iteration output divergence and token-level entropy convergence. (a)** Mean Jensen–Shannon divergence (in bits) between the prediction at each unrolled position and the final prediction, evaluated at the first answer position of ARC-Challenge questions. Shaded bands alternate to represent Ouro's four passes, with dots marking the completed pass ends. Internal layers exhibit severe prediction drift. **(b)** Next-token entropy at answer positions across recurrent steps for Huginn at $R = 32$ (zero-shot). The plot displays the median bounded by the quartile band. The recurrence settles rapidly, though the final distribution preserves significant uncertainty (dashed line denotes 2 nats).

This internal drift mathematically invalidates intra-iteration layers as guidance references. Our sweep of internal layers (layers 0, 8, 16, and 22/46 of the final pass at $\omega \in \{0.25, 0.5, 1.0\}$) revealed that extracting any internal state severely degrades ARC-Challenge performance, dropping by up to 20.6 points for Ouro-1.4B and 24.9 points for Ouro-2.6B below ordinary decoding. Conversely, the completed-iteration reference h_1 at identical strengths raises accuracy by 2.1 and 2.6 points. Unlike standard dense networks, where methods like DoLa (Chuang et al., 2024) successfully contrast intermediate layers against the final layer, looped architectures do not form trained exit points inside a pass, making intra-iteration states unstable for contrastive guidance.

Finally, observing this output convergence also explains the robustness of reduced-depth settings. Figure A8(b) traces Huginn's recurrence explicitly at the token level. The median next-token entropy over answer positions drops rapidly from 4.1 nats after the first step and effectively settles at 2.7 nats by step 16. It remains flat through step 32. Because the recurrence physically settles by the middle of its loop, halving Huginn's iterations costs its unguided seven-benchmark mean as little as 0.17 points (Section 4.3). Importantly, even upon settling, 67% of the token positions maintain an entropy above 2 nats (roughly an even choice among seven tokens). The recurrence converges on a distribution that still preserves meaningful ambiguity, which contrastive guidance resolves.

F Re-ranking Close Decisions

This appendix expands upon Section 5.2. We first detail how guidance gains heavily concentrate on the model's most uncertain decisions, including a step-by-step case study. We then geometrically decompose the logit update, demonstrating that LoopCD-Hidden fundamentally acts as an orthogonal re-ranking mechanism, while the remaining parallel component dictates temperature sharpening.

F.1 Concentration of guidance gains on uncertain decisions

Guidance effectively functions as a tie-breaker rather than a blunt overriding mechanism. It alters a final decision only when the injected contrast exceeds the probability margin between the top two candidate options. Consequently, confident predictions remain unperturbed, while undecided predictions shift.

Table A13 stratifies the evaluation questions into five equal-sized groups (fifths) based on the margin between the final prediction's top two options. Across every evaluated model, the least confident fifth commands the vast majority of the guidance gain, whereas the most confident fifth sees virtually zero impact. HellaSwag exhibits an even sharper concentration than ARC-Challenge. For instance, the least confident fifth yields +3.7 points for Ouro-1.4B and +7.3 for Ouro-2.6B, with no other fifth contributing more than +0.5. Ultimately, guidance flips very few answers overall (between 3.9% and 14.9%). The reported net gains represent a targeted excess of fixed answers over broken ones, concentrated on close decisions.

		Gain on each fifth, closest to farthest (points)					Answers changed	Net gain
Model	Benchmark	1	2	3	4	5		
Ouro-1.4B	ARC-Challenge	+6.41	+3.85	+0.43	0.00	0.00	6.6%	+2.1
Ouro-2.6B	ARC-Challenge	+7.26	+3.85	+1.28	+0.43	0.00	5.8%	+2.6
Huginn-0125	ARC-Challenge	+7.26	+4.70	+2.13	+0.85	+0.43	14.9%	+3.1
Parcae-1.3B	ARC-Challenge	+13.25	+5.56	-0.43	-0.85	0.00	8.8%	+3.5
Ouro-1.4B	HellaSwag	+3.69	+0.10	0.00	0.00	0.00	3.9%	+0.76
Ouro-2.6B	HellaSwag	+7.27	+0.45	+0.10	0.00	0.00	4.2%	+1.56

Table A13: **Guidance gains concentrate exclusively on the model's least confident predictions.** Evaluation of LoopCD-Logits at $\omega = 0.5$ with reference h_1. Questions are sorted into fifths based on the probability gap between the final prediction's top two options (the first fifth represents the least confident). The final two columns report the total proportion of changed answers and the net accuracy delta. Data represents ARC-Challenge (25-shot for Ouro and Parcae, zero-shot for Huginn at $R = 32$) and HellaSwag (10-shot).

To physically illustrate how these borderline decisions flip, Table A14 traces a single ARC-Challenge question through Ouro-1.4B's recurrent passes. The question asks which natural disaster leaves a narrow path of destruction through a forest. Across the early passes, the score for "tornado" rises significantly faster than for "earthquake" (+0.90 versus +0.32). However, at the final pass, "earthquake" still narrowly leads by a margin of less than 0.01. By applying guidance, LoopCD extrapolates this existing momentum, lifting "tornado" into first place. The guidance mechanism introduces no external knowledge. It simply reads the directional momentum from the reference state and continues shifting the options along that axis.

Option	After the first pass, alone	Final prediction	Guided, $\omega = 0.5$
a flood	-1.500	-0.770	-0.934
a tornado (correct)	-1.113	-0.209	**-0.170**
a hurricane	-0.969	-0.258	-0.247
an earthquake	-0.525	**-0.201**	-0.199

Table A14: **Trajectory of a re-ranked borderline decision.** Log-likelihood scores (per character) for Ouro-1.4B on an ARC-Challenge question (25-shot). Scores are shown after the first pass alone, at the final unguided pass, and under LoopCD-Logits ($\omega = 0.5$, reference h_1). Bold text indicates the leading option. Guidance extrapolates the momentum established during early passes to flip a close decision.

F.2 Geometric equivalence of LoopCD-Hidden and orthogonal re-ranking

As outlined in Section 5.2, a logit-space contrast vector can be mathematically decomposed into two orthogonal components: a part parallel to the final logits (which rescales the distribution like a temperature change) and a part orthogonal to the final logits (which re-ranks the relative standing of tokens).

Crucially, in architectures where the entire stack loops, LoopCD-Hidden behaves almost identically to the orthogonal re-ranking component of LoopCD-Logits.

In Ouro, where only normalization separates the recurrent loop from the head, empirical measurements confirm this equivalence. We evaluate the final answers (verdicts) produced by three distinct guidance applications: LoopCD-Hidden, the full logit update (applying both orthogonal and parallel components), and the isolated logit re-ranking update (applying strictly the orthogonal component).

On HellaSwag at $\omega = 0.5$, LoopCD-Hidden diverges from the pure logit re-ranking component on merely 205 of the 10,042 evaluation verdicts for Ouro-1.4B (136 for Ouro-2.6B). In stark contrast, it diverges from the full logit update on a much larger 508 verdicts (357 for Ouro-2.6B). Figure A9 illustrates how effectively the hidden-state application replicates the isolated orthogonal behavior.

The update directions show the same pattern in logit space (Figure A10). At $\omega = 0.25$, the hidden-state update has a mean cosine similarity of 0.96 with the orthogonal re-ranking update for both Ouro models, far above its similarity to the full logit update (0.25 for Ouro-1.4B and 0.44 for Ouro-2.6B).

For Huginn, where a coda separates the loop from the head, the dynamics shift slightly. When using its optimal sixth state as a reference, the hidden-state update still closely tracks the re-ranking component (cosine similarity 0.86 at $\omega = 0.5$). However, when forced to use the first state, alignment drops to 0.62 at the same strength. This drop occurs because Huginn's first hidden state is dominated by initialization noise (Appendix E.1), injecting random directional shifts rather than a coherent re-ranking signal.

F.3 The double-edged effect of temperature sharpening

Because LoopCD-Hidden effectively mirrors orthogonal re-ranking, the functional difference between the hidden and logit forms is precisely the parallel component that the logit form retains: temperature sharpening.

The utility of this sharpening component is strictly benchmark-dependent. On HellaSwag, aggressive sharpening degrades performance. The full logit update at $\omega = 0.5$ yields +0.76 points for

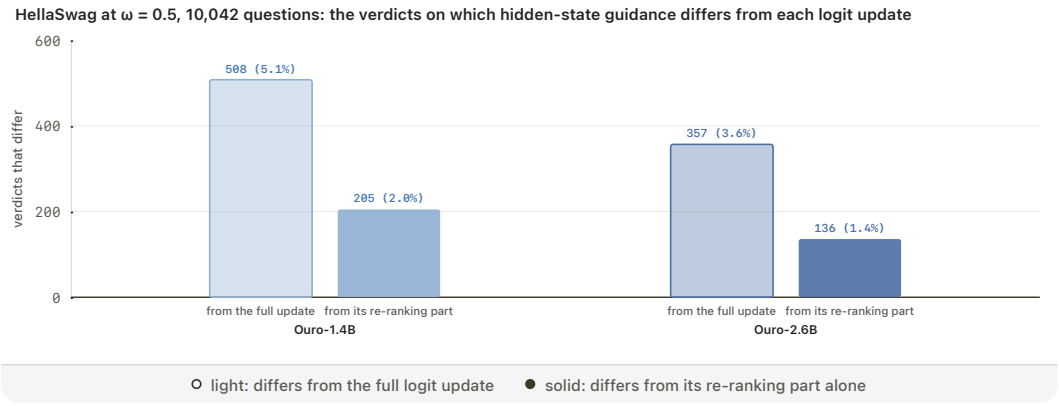

Figure A9: **LoopCD-Hidden replicates the verdicts of orthogonal logit re-ranking.** Evaluation on 10,042 HellaSwag questions at $\omega = 0.5$ with reference h_1. The bars display the number of discordant verdicts where LoopCD-Hidden differs from the full logit update (which applies both temperature scaling and re-ranking, light bars) versus where it differs from the isolated logit re-ranking component (which applies strictly the orthogonal shift, solid bars).

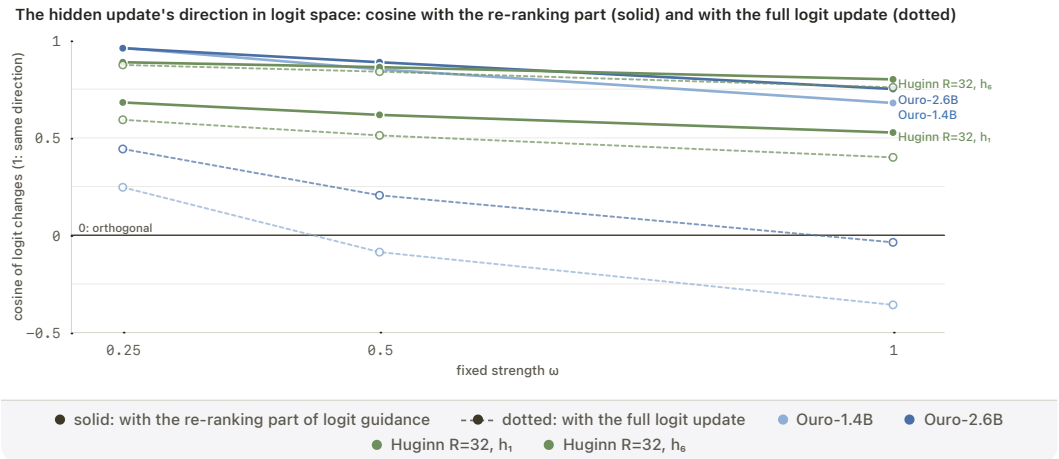

Figure A10: **Hidden-state updates closely align with the orthogonal re-ranking direction.** Cosine similarity measured in mean-centered logit space between the LoopCD-Hidden update and two logit counterparts: the isolated re-ranking component (solid lines) and the full logit update (dotted lines). Values represent the per-position mean over 300 HellaSwag documents. Parcae-1.3B is omitted due to non-deterministic `bf16` execution variance at this sequence length.

Ouro-1.4B, whereas isolating its re-ranking component yields +1.72. Because the hidden-state form naturally lacks this detrimental sharpening, it achieves double the gain (+1.73 points). Conversely, on ARC-Challenge, sharpening provides a distinct advantage. The pure re-ranking component requires a much larger strength ($\omega = 1.3$) to reach its peak gain of +2.65, whereas the full logit update peaks efficiently at $\omega = 0.9$ with +2.47.

Consequently, for fully recurrent models lacking a coda, selecting between LoopCD-Logits and LoopCD-Hidden is effectively a dataset-dependent choice regarding whether temperature sharpening is beneficial, typically swaying net accuracy by one or two percentage points. Across our broader seven-benchmark suite, this dynamic holds consistently: LoopCD-Hidden excels on HellaSwag but trails LoopCD-Logits on ARC-Challenge, ARC-Easy, SciQ, and MMLU.

G Guidance Strength Dynamics

This appendix sweeps the fixed and adaptive guidance strengths, providing benchmark-level resolution for generation tasks and detailing the spatial distribution of updates across confidence fifths to explain the overshoot phenomenon at high guidance strengths.

G.1 Fixed and adaptive strength sweeps

Multiple-choice scoring tolerates broad guidance strength, whereas autoregressive generation requires smaller values to prevent compounding errors. For multiple-choice scoring, accuracy gains remain positive across a wide strength band peaking near $\omega = 0.5$ (Figure A11). In contrast, autoregressive generation requires a narrower operating window ($\omega \in [0.2, 0.3]$), as early token shifts compound across generated prefixes and cause steep declines at larger strengths. Moving toward the weaker reference ($\omega < 0$) universally degrades accuracy (Appendix G.2).

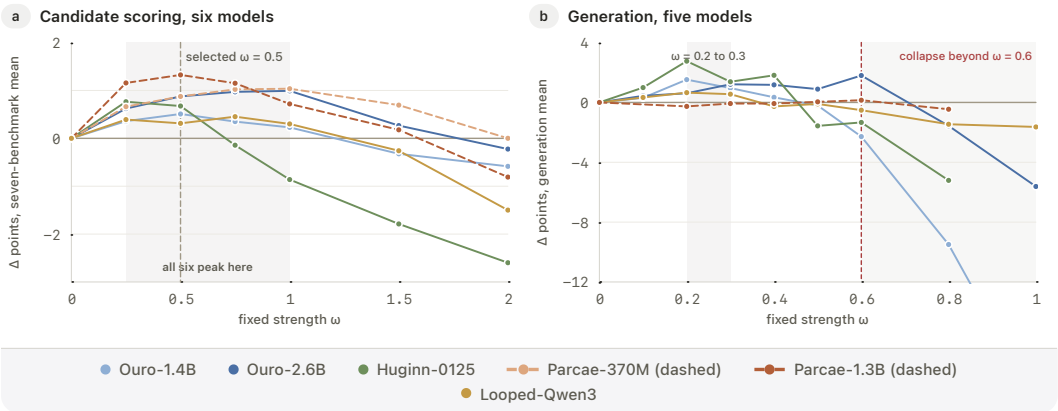

Figure A11: **Multiple-choice scoring tolerates about twice the fixed strength that generation does. (a)** Seven-benchmark mean change against the fixed logit strength, six models. The band spans the strengths at which they peak, and the dashed line marks the selected $\omega = 0.5$. **(b)** Generation mean change, five models. The band marks the selected 0.2 to 0.3, and beyond $\omega \approx 0.6$ the curves fall steeply.

Adaptive margin gating stabilizes decoding at large strengths by scaling guidance with prediction uncertainty. By modulating strength via the top-two probability margin (Eq. (4)), adaptive guidance applies maximum strength $\omega_{\max}$ only to contested tokens and smoothly withholds updates on settled decisions. This margin gating prevents high-confidence drift, maintaining positive accuracy gains at caps where fixed strength overshoots (Figure A12; Appendix G.3).

G.2 Strength boundaries: generation collapse and negative interpolation

Figure A13 disaggregates the generation sweeps from Figure A11(b) across individual benchmarks to illustrate the upper bounds of beneficial guidance. The operating window for autoregressive generation is uniformly narrow, and the degradation beyond this optimal positive range is catastrophic.



Figure A12: **The adaptive rule keeps its gain over a wider range of strength than a fixed value.** **(a)** Seven-benchmark mean change against the adaptive cap $\omega_{\max}$, six models. The band spans the caps at which they peak. **(b)** Mean over the six models, the fixed rule against its strength (dashed) and the adaptive rule against its cap (solid). The band is where the adaptive mean is still above the baseline and the fixed mean is not.

For instance, on GSM8K, Ouro-1.4B loses 8 points at $\omega = 0.8$ and plummets by 20 points at $\omega = 1.0$. HumanEval exhibits similarly steep declines of 10 and 22 points at the same respective strengths.

While generation tasks expose the strict upper bounds of positive guidance, evaluating multiple-choice scoring at negative strengths exposes the vector's lower bounds. Negative strengths explicitly interpolate backward along the contrast vector toward the weaker reference state. As mathematically expected, applying a negative ω uniformly degrades the seven-benchmark multiple-choice mean across all models at every value, predictably converging entirely to the standalone reference accuracy at $\omega = -1.0$ (Figure A5(a)). This symmetric behavior validates that the isolated contrast vector strictly encodes the direction of model improvement.

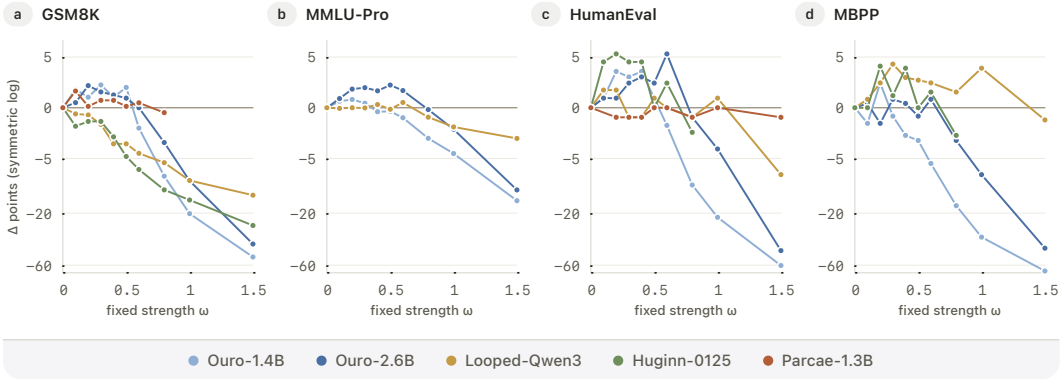

Figure A13: **Generation tasks exhibit a strictly narrow operating window across all benchmarks.** The change from ordinary decoding against the fixed logit strength for the five generation models on GSM8K, MMLU-Pro, HumanEval, and MBPP. A symmetric-log axis is used to preserve the frame amidst the catastrophic collapse beyond $\omega \approx 0.6$. Huginn's GSM8K curve reflects its chain-of-thought setting, and Parcae-1.3B was evaluated exclusively on GSM8K and HumanEval.

G.3 The trade-off between decision reach and overshoot

As shown in Appendix G.1, applying a large fixed strength overshoots and corrupts previously settled decisions. Table A15 quantifies this phenomenon on ARC-Challenge, illustrating the inherent trade-off between extending reach into undecided questions and overshooting established ones as

	Answers changed			Least confident fifth		Most confident fifth
Model	$\omega = 0.25$	0.5	1	$\omega = 0.5$	1	$\omega = 0.5$
Ouro-1.4B	3.8%	6.6%	9.0%	+6.4	+3.9	0.0
Huginn-0125	9.1%	14.9%	20.7%	+7.3	+5.1	+0.4
Parcae-1.3B	–	8.8%	–	+13.3	–	0.0

Table A15: **Update distribution across confidence fifths as guidance strength grows on ARC-Challenge.** The overall share of changed answers, and the specific accuracy change (in points) isolated within the least confident and most confident fifths at three fixed strengths ω. Evaluated under LoopCD-Logits. Ouro-1.4B and Parcae-1.3B use 25-shot prompting, while Huginn-0125 uses zero-shot at $R = 32$. A dash indicates a strength setting not evaluated during the per-question sweep for that specific model.

the fixed ω increases from 0.25 to 1.0. Questions are stratified into fifths based on the probability margin between the final prediction's two leading candidates (from least to most confident). Table A15 summarizes the answer-change proportions and the accuracy delta within each fifth across model families.

As ω grows, guidance forcefully modifies a progressively larger fraction of model verdicts. For Ouro-1.4B, the overall share of changed answers rises monotonically from 3.8% at $\omega = 0.25$ to 6.6% at $\omega = 0.5$ and 9.0% at $\omega = 1.0$. Huginn-0125 exhibits a parallel progression across a wider dynamic range, changing 9.1% of answers at $\omega = 0.25$, 14.9% at $\omega = 0.5$, and 20.7% at $\omega = 1.0$. Parcae-1.3B similarly modifies 8.8% of verdicts at $\omega = 0.5$. Across all architectures, larger coefficients extend the update's reach into decisions that sat much farther from the decision boundary in the unguided baseline.

However, analyzing the spatial distribution of these accuracy gains reveals why net performance consistently peaks at moderate strengths. At $\omega = 0.25$, accuracy gains are confined almost entirely to the least confident fifth, with negligible impact on remaining questions. At $\omega = 0.5$, guidance optimally improves this least confident fifth (+6.4 points for Ouro-1.4B, +7.3 points for Huginn) while gradually spreading gains into the second and third fifths.

Crucially, at $\omega = 1.0$, the accuracy within the least confident fifth actively recedes: Ouro-1.4B drops to +3.9 points, and Huginn drops to +5.1 points. Although $\omega = 1.0$ successfully forces updates in more confident fifths, it severely overshoots the optimal decision boundary for the borderline alternatives in the least confident fifth, aggressively flipping previously corrected predictions back into errors. The aggregate score universally peaks between $\omega = 0.25$ and $\omega = 1.0$ because the marginal benefit of reaching more confident fifths is eventually eclipsed by the mounting overshoot penalties on close decisions.

H Related Work

Looped Transformer and recurrent computing. Reusing Transformer layers was originally proposed to increase effective network depth without a proportional increase in parameter count (Dehghani et al., 2019). Recent paradigms have shifted the primary focus of recurrence toward continuous latent reasoning and test-time computation scaling (Hao et al., 2025; Chen et al., 2025; Kang et al., 2026; Du et al., 2026; Liu et al., 2026). By iteratively applying shared parameters, models can implicitly unroll reasoning steps in the latent space to solve complex logical tasks and generalize to longer sequences (Giannou et al., 2023; Yang et al., 2024; Fan et al., 2025; Saunshi et al., 2025; Zhu et al., 2025). Pre-trained architectures like Ouro (Zhu et al., 2026), Huginn (Geiping et al., 2025), and Parcae (Prairie et al., 2026) explicitly embed this recurrent trajectory into the language modeling objective. Extracting maximal reasoning performance from these trajectories often involves intervening during training, such as supervising latent steps with reasoning traces (Fan et al., 2026) and distributing reinforcement learning credit along the loop (Williams et al., 2026), or optimizing loop counts for parallel refinement during inference (Yang et al., 2026). LoopCD diverges from these structural modifications by capitalizing on the natural semantic divergence between early and final iterations of pre-trained loops, leveraging the trajectory to drive weak-to-strong extrapolation strictly at inference time.

Weak-to-strong contrastive decoding. Contrastive decoding (CD) improves generation by extrapolating an expert model's prediction away from a weaker reference, isolating and amplifying the expert's advanced capabilities (Li et al., 2023; O'Brien & Lewis, 2023). The core bottleneck of CD remains the acquisition of an appropriate weak reference. While context-aware decoding generates this reference by perturbing the input prompt (Shi et al., 2024; Tang et al., 2026), DoLa (Chuang et al., 2024) and Adaptive Layer-Wise CD (Zhou et al., 2025) bypass external modifications by extracting reference logits from intermediate spatial layers within the same dense model. However, spatial layer contrast fails in recurrent architectures. As demonstrated in our analysis, internal layers within an uncompleted recurrent loop drift drastically and do not constitute valid predictive states. LoopCD therefore redefines the contrastive axis from spatial depth to temporal iterations of the identical parameter block. This temporal contrast guarantees that the weak reference represents a structurally valid, fully projected prediction that perfectly shares the vocabulary and feature space of the final state, eliminating the architectural mismatch inherent in multi-model or spatial-layer CD.

Guidance in diffusion models. Guidance is a technique where a diffusion model's output is slightly adjusted at inference time, and has become a critical component of modern image generation models. Classifier-free guidance constructs a guidance vector using the difference between a conditional and an unconditional model pass (Ho & Salimans, 2021). Similarly, autoguidance uses the difference between a weak and a strong model, where the weak and strong model must share similar dynamics (in the sense that they make the same sorts of errors) (Karras et al., 2024). Although generally applied to diffusion models, both CFG and autoguidance can be applied to autoregressive models over tokens (Sanchez et al., 2024; Li et al., 2023). LoopCD works similarly to autoguidance by using the difference between a strong (deep) and weak (shallow) model pass to guide the final output.