OneiroDraft: Reading the Latent Dreams for Speculative Decoding

Anonymous authors
Paper under double-blind review

Abstract

Speculative decoding accelerates autoregressive generation by pairing a lightweight drafter with parallel target-model verification. Its effectiveness depends on producing accurate candidate tokens at sufficiently low drafting cost. Existing approaches make different compromises between these objectives: independent draft models incur separate model computation and may deviate from the target model, while auxiliary prediction heads remain closely coupled to the target but are typically most effective over short prediction horizons. We explore an alternative based on the target model's intermediate hidden states, which already encode rich contextual information and can be extrapolated through lightweight transitions instead of repeated full-model computation. To this end, we introduce **OneiroDraft**, a recurrent latent drafter that adapts a small contiguous interval of the pretrained target Transformer into a shared transition for predicting future hidden states and converting them into candidate tokens. To improve prediction over multiple steps, we first train the transition and readout modules separately and then jointly optimize them over recurrent rollouts, aligning state prediction with token generation. During inference, OneiroDraft recurrently generates drafts using the learned transition module and combines them with contextual candidate reuse to reduce drafting overhead, while a configurable verification criterion controls the trade-off between strict target matching and candidate acceptance. On Qwen3-8B, recurrent latent drafting alone achieves $1.20\times$–$1.27\times$ wall-clock speedup, while the Hybrid decoder that combines it with contextual reuse reaches $1.59\times$–$1.70\times$ wall-clock speedup and $1.60\times$–$1.70\times$ throughput improvement over paired greedy decoding. These results show that pretrained target computation can be successfully repurposed into an inexpensive recurrent proposal mechanism: the adapted target interval can advance hidden representations along the target model's future trajectory and provide useful speculative candidates without repeatedly executing the complete target model.

1 Introduction

Autoregressive decoding requires a full forward pass through the target model for every generated token, making long-form generation increasingly expensive. Speculative decoding reduces this cost by using cheaper auxiliary computation to propose multiple future tokens and verifying them in parallel with the target model (Leviathan et al., 2023; Chen et al., 2023). Its practical speedup depends on a basic trade-off: proposals must be accurate enough to be accepted, yet inexpensive enough that drafting costs less than the target computation it replaces. Existing approaches instantiate this trade-off with independent draft models, lightweight multi-token prediction heads, partial target execution, and recurrent feature predictors (Cai et al., 2024; Gloeckle et al., 2024; Cheng et al., 2024; Li et al., 2024a; 2026). Despite their architectural differences, learned drafters must ultimately advance beyond the committed prefix while remaining substantially cheaper than the target model.

We take a different route by reusing computation already learned inside the pretrained target model. A stack of Transformer layers normally transforms representations along model depth; our key idea is to adapt this computation to advance an intermediate representation across future token positions instead. Starting from a target hidden state h_t, we copy a small contiguous interval of target Transformer

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.

layers and adapt it into a shared transition T_ϕ,

$$h_t \to \hat{h}_{t+1} \to \hat{h}_{t+2} \to \cdots,$$

where each $\hat{h}_{t+k}$ approximates the representation the target model would produce at the same boundary and future position. Based on this formulation, we introduce **OneiroDraft**, a recurrent latent-state drafter that repeatedly applies the same transition initialized from target layers and reads candidate tokens from the resulting latent trajectory. Figure 1 summarizes this shift from computation along model depth to cross-position speculative drafting.

Recurrent latent drafting introduces two additional considerations. First, later predictions depend on states produced by earlier speculative steps, making recurrent rollout more difficult than isolated next-state prediction. We therefore use a training procedure with staged initialization followed by joint calibration that first establishes state transition and token readout and then calibrates their composition under recurrent rollout. Second, neural drafting is unnecessary when the committed context already contains a reusable continuation. We therefore combine ONEIRODRAFT with contextual candidate reuse (He et al., 2024b; Hu et al., 2025), using inexpensive lookup when available and recurrent latent drafting otherwise.

Our experiments separately evaluate recurrent latent drafting, recurrent calibration, and the complete Hybrid decoder. With Qwen3-8B, recurrent latent drafting alone achieves $1.20\times$–$1.27\times$ **wall-clock speedup**, demonstrating that the learned cross-position transition provides net end-to-end acceleration without contextual reuse. Recurrent joint calibration consistently improves draft acceptance and the decoding speed of latent drafting alone, while combining latent drafting with contextual reuse increases wall-clock speedup to $1.59\times$–$1.70\times$. Answer accuracy remains broadly comparable to paired greedy decoding, with modest variations across tasks in either direction.

Our contributions are:

- We introduce a **recurrent drafting formulation built from target Transformer layers** that repurposes a contiguous interval of the pretrained target Transformer from representation transformation along model depth into a shared transition across speculative positions.
- We develop a **recurrent training procedure** that improves latent prediction over recurrent rollouts by calibrating the transition and readout together.
- We show that **recurrent latent drafting provides end-to-end acceleration on its own**, that recurrent calibration consistently improves the learned drafter, and that contextual reuse provides additional speedup when reusable continuations are available.

2 RELATED WORK

Speculative drafting. Speculative decoding accelerates autoregressive generation by replacing several target-model decoding steps with inexpensive proposal generation followed by parallel verification (Stern et al., 2018; Leviathan et al., 2023; Chen et al., 2023). Existing methods differ mainly in how proposals are constructed. Independent draft models use a smaller autoregressive network, with DistillSpec improving agreement between the drafter and target through distillation (Zhou et al., 2024). Other methods reuse representations from the target model: Medusa and multi-token prediction attach parallel prediction heads, while Hydra introduces dependencies between successive draft heads (Cai et al., 2024; Gloeckle et al., 2024; Ankner et al., 2024). Methods based on self-speculation, such as Draft & Verify and LayerSkip, reduce drafting cost by reusing partial target computation (Zhang et al., 2024; Elhoushi et al., 2024). These approaches expose a recurring trade-off: inexpensive parallel prediction limits computation per future position, whereas modeling dependencies among successive proposals generally introduces additional sequential work. Recurrent drafting provides another point in this design space by propagating a compact representation across speculative positions.

Recurrent drafting. Recurrent drafters allow later proposals to depend on representations produced at earlier speculative positions. ReDrafter uses a recurrent auxiliary model conditioned on target hidden states (Cheng et al., 2024). EAGLE instead autoregressively predicts target features, while EAGLE-2 augments this formulation with draft trees adapted to the context (Li et al., 2024a;b). EAGLE-3 moves away from explicit feature reconstruction and directly optimizes token prediction using features from multiple target layers (Li et al., 2026). FastMTP recursively reuses a prediction

Table 1: Comparison of representative proposal mechanisms coupled to the target model. **Sequential dep.** indicates whether later proposals depend on earlier speculative predictions; **Target reps.** indicates direct use of target-model representations; **Shared** indicates reuse of the same proposal operator across speculative positions; and **Copied target block** indicates that a copied target Transformer interval itself is adapted as the transition.

Method	Sequential dep.	Target reps.	Shared	Early exit	Copied target block
Medusa heads (Cai et al., 2024)	×	✓	×	×	×
MTP heads (Gloeckle et al., 2024)	×	✓	×	×	×
Hydra (Ankner et al., 2024)	✓	✓	×	×	×
FastMTP (Cai et al., 2025)	✓	✓	✓	×	×
ReDrafter (Cheng et al., 2024)	✓	✓	✓	×	×
EAGLE-3 (Li et al., 2026)	✓	✓	✓	×	×
Make Every Draft Count (Chen et al., 2026)	✓	✓	✓	×	×
Draft & Verify (Zhang et al., 2024)	✓	✓	✓	✓	×
LayerSkip (Elhoushi et al., 2024)	✓	✓	✓	✓	×
ONEIRODRAFT (ours)	✓	✓	✓	×	✓

head shared across positions (Cai et al., 2025), and Make Every Draft Count performs autoregression over hidden states while delaying token information injection so that speculative computation can be reused after verification failures (Chen et al., 2026). ONEIRODRAFT differs primarily in the origin of the recurrent operator: instead of introducing a separate recurrent predictor or prediction head, it adapts a contiguous interval copied from the pretrained target Transformer into the transition across speculative positions.

Recursive Transformers. Parameter sharing across repeated Transformer computation has also been explored independently of speculative decoding. Universal Transformers repeatedly apply shared transformations for iterative representation refinement, while Relaxed Recursive Transformers convert pretrained Transformers into models that reuse blocks across effective depth with lightweight adaptation specific to each layer (Dehghani et al., 2018; Bae et al., 2025). In these models, recurrence acts primarily along the depth or refinement axis. ONEIRODRAFT instead reuses the copied Transformer interval across future token positions, changing the role of the pretrained computation from depth-wise transformation to cross-position drafting.

Training recurrent trajectories and proposal reuse. Recurrent drafting introduces a train–inference mismatch because later predictions are conditioned on states produced by earlier draft steps, allowing errors to accumulate along the rollout. Distillation improves agreement between the drafter and target (Zhou et al., 2024), HASS explicitly addresses mismatches in training context and objective (Zhang et al., 2025), and methods such as FastMTP and EAGLE-3 train prediction modules under recursive drafting over multiple steps (Cai et al., 2025; Li et al., 2026). Work on latent prediction further studies error accumulation over longer recurrent trajectories (Shi et al., 2026), while recent analyses identify hidden-state degradation and attention drift as additional sources of error over longer drafting horizons (Liu et al., 2026; Eldenk et al., 2026). ONEIRODRAFT first initializes transition and readout separately and then jointly calibrates them on non-detached recurrent trajectories. Separately, retrieval methods can avoid neural proposal computation altogether: REST retrieves candidate continuations from stored text, while SAM Decoding uses suffix matching and can be combined with learned drafting methods (He et al., 2024b; Hu et al., 2025). ONEIRODRAFT uses contextual reuse when a suitable continuation is available and otherwise falls back to recurrent latent drafting.

3 METHOD

ONEIRODRAFT formulates speculative drafting as recurrent extrapolation in the target model's hidden space. We copy a small contiguous interval of the pretrained target Transformer and adapt it from representation transformation along model depth into a shared transition across future token positions. Repeated application of this transition produces a short latent trajectory in the target model's hidden space, from which candidate tokens are read out and verified by the complete target model. Figure 1 illustrates how pretrained computation that originally operates along model depth is repurposed into a recurrent transition across speculative token positions.

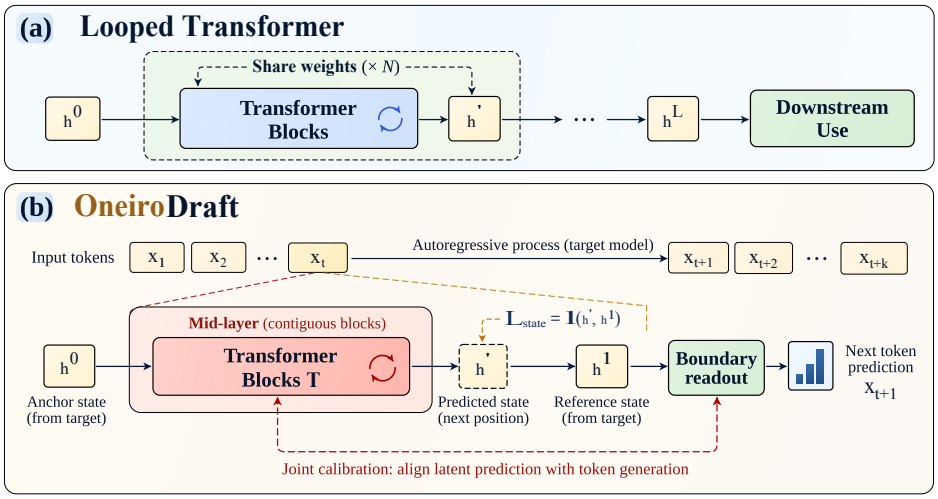

Figure 1: **From depth-wise recurrence to cross-position latent drafting. Top:** A looped Transformer reuses shared computation along the depth or refinement axis. **Bottom:** ONEIRODRAFT copies a contiguous interval of the pretrained target Transformer and adapts it into a shared transition across future token positions. Starting from h_t, the transition predicts future states such as $\hat{h}_{t+1}$, whose token-space readouts form speculative candidates for parallel target verification.

3.1 PROBLEM FORMULATION

Let $x_{\leq t} = (x_1, \ldots, x_t)$ denote the committed prefix, and let F_θ be the complete target model with next-token distribution $p_\theta(\cdot \mid x_{\leq t})$. We select an intermediate boundary b and denote the corresponding target representation by

$$h_{\leq t} = H_\theta^{(b)}(x_{\leq t}) = (h_1, \ldots, h_t), \qquad h_i \in \mathbb{R}^d, \tag{1}$$

where $H_\theta^{(b)}$ denotes target computation up to boundary b.

Our goal is to advance this representation across several future positions without repeatedly executing F_θ. For a latent sequence S, we write $\tau_\phi(S) = \text{last}[T_\phi(S)]$. Starting from the prefix states $h_{\leq t}$ computed by the target model, we recursively predict

$$\hat{h}_{t+k} = \tau_\phi\left(h_{\leq t} \oplus \hat{h}_{t+1:t+k-1}\right), \qquad k = 1, \ldots, K, \tag{2}$$

where $\oplus$ denotes sequence concatenation and the suffix of predicted states is empty for $k = 1$. The desired rollout satisfies $\hat{h}_{t+k} \approx h_{t+k}$ at each speculative position. Thus, the recurrent state is the hidden representation itself rather than a predicted token.

3.2 CROSS-POSITION DRAFTING FROM TARGET LAYERS

The key architectural choice is the provenance of T_ϕ. Let $\mathcal{I}$ denote a small contiguous interval of pretrained target layers with parameters $\theta_\mathcal{I}$. We initialize

$$\phi^{(0)} = \text{Copy}\left(\theta_\mathcal{I}\right), \tag{3}$$

after which the copied module is optimized independently and maintains its own drafting cache. In the original target model, these layers transform representations along model depth; in ONEIRODRAFT, they are adapted to advance the selected representation across token positions. Further motivation for this initialization from target layers and a controlled comparison against an otherwise identical random initialization are provided in Appendices A.2 and A.3.

A lightweight readout B_ψ converts each predicted state into a token distribution:

$$q_\psi(v \mid h) = \text{softmax}(B_\psi(h))_v, \qquad d_{t,k} = \arg\max_v q_\psi(v \mid \hat{h}_{t+k}). \tag{4}$$

The hidden state $\hat{h}_{t+k}$ remains the recurrent object, while $d_{t,k}$ is used only for target verification. The exact parameterization of B_ψ is given in Appendix B.1.

Following the target model's next-token convention, the complete target model supplies the anchor $a_t = \arg\max_v p_\theta(v \mid x_{\le t})$. The first predicted state $\hat{h}_{t+1}$ produces the first additional draft token, and a neural rollout of depth K yields the draft sequence $(d_{t,1}, \ldots, d_{t,K})$.

3.3 Training the Cross-Position Transition

Recurrent drafting creates a train–inference mismatch because later transition steps consume predicted rather than target-computed states. We therefore use staged initialization followed by joint recurrent calibration.

Staged initialization. We first train T_ϕ to predict the next target state:

$$\mathcal{L}_{\text{step}} = \mathbb{E}_{x,t}\left[\frac{1}{d}\left\|\tau_\phi(h_{\le t}) - \text{sg}(h_{t+1})\right\|_2^2\right], \tag{5}$$

where $\text{sg}(\cdot)$ denotes stop-gradient. We then expose T_ϕ to short recurrent rollouts in which predicted states are reused as subsequent inputs but detached before reuse. This exposes the transition to the imperfect latent contexts encountered at inference while preventing later-step losses from propagating through earlier predictions.

Separately, B_ψ is initialized on target-computed hidden states using supervision from ground-truth next tokens together with distillation from the adapted target distribution, which is held fixed during readout initialization (Zhou et al., 2024). The detached recurrent objective, readout initialization objective, and complete optimization schedule are given in Appendix B.1.

Joint recurrent calibration. After staged initialization, we freeze F_θ and jointly optimize T_ϕ and B_ψ on the recurrence in Eq. 2 without detachment. For $K \ge 2$, we constrain both the predicted latent trajectory and the token distribution decoded from it:

$$\mathcal{L}_{\text{state}} = \mathbb{E}_{x,t}\left[\frac{1}{K-1}\sum_{k=2}^{K}\ell_\beta\left(\hat{h}_{t+k}, \text{sg}(h_{t+k})\right)\right], \tag{6}$$

$$\mathcal{L}_{\text{dist}} = \mathbb{E}_{x,t}\left[\frac{1}{K}\sum_{k=1}^{K} D_{\text{KL}}\left(\text{sg}[p_\theta(\cdot \mid x_{\le t+k})] \,\|\, q_\psi(\cdot \mid \hat{h}_{t+k})\right)\right], \tag{7}$$

where ℓ_β denotes the element-wise smooth-L_1 state distance defined in Appendix B.1. The first predicted state is already supervised by $\mathcal{L}_{\text{step}}$, so the additional state term for later rollout steps begins at $k = 2$.

The complete joint calibration objective is

$$\mathcal{L}_{\text{joint}} = \alpha\mathcal{L}_{\text{step}} + w(s)\left(\mathcal{L}_{\text{state}} + \gamma\mathcal{L}_{\text{dist}}\right). \tag{8}$$

Here, s denotes the calibration step, α and γ balance the one-step, state, and distribution objectives, and $w(s)$ controls the contribution of recurrent supervision during calibration. Stage lengths and learning rates are summarized in Appendix B.1. Because the recurrent states are no longer detached, losses at later speculative positions can also update the earlier transition steps that produced them. The optimization settings for each stage and auxiliary loss definitions are detailed in Appendix B.1, while the observed training and validation trajectories are reported in Appendix B.2.

3.4 Speculative Inference and Re-Anchoring

At each decoding round, the target model provides the committed latent context $h_{\le t}$ and the anchor a_t generated by the target model. The proposal mechanism then produces a candidate sequence $C_t = (c_{t,1}, \ldots, c_{t,K_t})$. For neural drafting, Eq. 2 constructs the predicted latent trajectory and Eq. 4 yields $c_{t,j} = d_{t,j}$; when contextual reuse is available, the candidates are instead supplied directly from the committed context (He et al., 2024b; Hu et al., 2025).

The complete target model evaluates the anchor and candidate sequence in one causal forward pass (Leviathan et al., 2023; Chen et al., 2023). For the j-th candidate, define

$$\pi_{t,j}(v) = p_\theta\left(v \mid x_{\le t}, a_t, c_{t,<j}\right), \qquad j = 1, \dots, K_t. \tag{9}$$

Under strict verification, the number of accepted candidates is

$$m_t = \max\left\{m \in \{0, \dots, K_t\} : c_{t,j} = \arg\max_v \pi_{t,j}(v),\ \forall j \le m\right\}. \tag{10}$$

The committed prefix is therefore updated as

$$x_{\le t'} = x_{\le t} \oplus \left(a_t, c_{t,1:m_t}\right), \qquad t' = t + 1 + m_t. \tag{11}$$

The relaxed target-support criterion used in additional experiments is defined in Appendix B.3.

Predicted latent states are temporary and are not propagated across speculative rounds. After verification, the recurrent context is reset to the states computed by the target model for the newly committed prefix,

$$h_{\le t'} = H_\theta^{(b)}\left(x_{\le t'}\right). \tag{12}$$

Thus, recurrent prediction errors can accumulate within a speculative block but do not propagate indefinitely across generation. Detailed cache synchronization and rollback behavior are described in Appendix B.4.

4 Experiments

We evaluate the end-to-end performance of OneiroDraft and then isolate the sources of its acceleration. We first report the complete Hybrid decoder, then disentangle recurrent latent drafting from contextual reuse, study the effect of recurrent joint calibration, and finally analyze sensitivity to the verification threshold, speculation start position, and response length.

4.1 Experimental Setup

Training. We train OneiroDraft on MetaMathQA (Yu et al., 2024) using Qwen3-4B and Qwen3-8B as target models. Backbone weights remain frozen, with trainable adaptation on the target side implemented through LoRA (Hu et al., 2021). Training follows the training procedure described in Sec. 3.3: we first establish state transition and token readout through one-step, detached recurrent, and readout initialization stages, and then freeze the resulting target while jointly calibrating T_ϕ and B_ψ on recurrent rollouts. We evaluate the model after 4,000 joint calibration updates and, for Qwen3-8B, also after Extended calibration with 20,000 total joint calibration updates. Stage lengths and learning-rate settings are provided in Appendix B.1.

Models and benchmarks. We evaluate Qwen3-4B and Qwen3-8B (Yang et al., 2025) on seven mathematical reasoning benchmarks: MATH-500 (Hendrycks et al., 2021; Lightman et al., 2024), AIME 2024, AIME 2025, AIME 2026, AMC 2023, Minerva/OCW (Lewkowycz et al., 2022), and OlympiadBench (He et al., 2024a). The evaluation sets contain 500, 30, 30, 30, 40, 272, and 675 examples, respectively. Unless otherwise stated, results are reported as mean $\pm$ sample standard deviation over training seeds 42, 43, and 44.

Decoding protocol. Unless otherwise stated, inference uses BF16 precision, batch size 1, a maximum prompt length of 4096 tokens, and a maximum response length of 1024 tokens. The recurrent transition proposes up to two neural draft tokens per speculative round, while contextual lookup may provide up to eight candidates. Speculation begins after 32 generated tokens and uses strict verification with $\lambda = 1.00$. Each speculative run is paired with greedy decoding using the same target checkpoint; speedups are first computed relative to this paired baseline and then aggregated across seeds. We report both wall-clock speedup and token-throughput speedup.

Under strict verification, a candidate is committed only when it matches the target argmax produced by the corresponding verification forward pass. Blockwise verification and token-by-token greedy decoding can nevertheless produce different output sequences because their BF16 GPU execution paths are not necessarily identical. We therefore report final-answer accuracy separately from decoding speed and do not interpret strict verification as sequence-level equivalence to an independently executed greedy run.

Table 2: Qwen3-8B decoding results after Extended calibration. Values are mean ± sample standard deviation over seeds 42, 43, and 44.

Dataset	Accuracy (%)		Speedup over Greedy (×)	
	Greedy	Hybrid	Wall-clock	Throughput
MATH-500	66.60 ± 0.92	67.27 ± 0.12	1.63 ± 0.01	1.63 ± 0.01
AIME 2024	4.44 ± 1.92	3.33 ± 0.00	1.70 ± 0.05	1.70 ± 0.02
AIME 2025	4.44 ± 1.92	4.44 ± 1.92	1.65 ± 0.01	1.65 ± 0.01
AIME 2026	1.11 ± 1.92	1.11 ± 1.92	1.64 ± 0.05	1.64 ± 0.04
AMC 2023	27.50 ± 4.33	29.17 ± 1.44	1.67 ± 0.04	1.66 ± 0.03
Minerva/OCW	25.61 ± 0.42	26.72 ± 1.39	1.59 ± 0.02	1.60 ± 0.01
OlympiadBench	23.51 ± 0.75	23.11 ± 0.90	1.66 ± 0.02	1.66 ± 0.02

Table 3: Ablation of proposal sources on Qwen3-8B. Wall-clock speedups are measured relative to paired greedy decoding and reported as mean ± sample standard deviation over three seeds. The highest mean speedup in each row is boldfaced.

Dataset	Latent-only (×)	Lookup-only (×)	Hybrid (×)
MATH-500	1.27 ± 0.01	1.51 ± 0.01	**1.63** ± 0.01
AIME 2024	1.21 ± 0.04	1.64 ± 0.01	**1.70** ± 0.05
AIME 2025	1.20 ± 0.01	1.59 ± 0.02	**1.65** ± 0.01
AIME 2026	1.22 ± 0.01	1.56 ± 0.05	**1.64** ± 0.05
AMC 2023	1.27 ± 0.01	1.56 ± 0.04	**1.67** ± 0.04
Minerva/OCW	1.20 ± 0.01	1.50 ± 0.01	**1.59** ± 0.02
OlympiadBench	1.23 ± 0.01	1.56 ± 0.01	**1.66** ± 0.02

4.2 Decoding Performance

We first evaluate the complete Hybrid decoder, which combines recurrent latent drafting with contextual candidate reuse. For the complete decoder evaluation, we use Qwen3-8B after staged initialization followed by 20,000 recurrent joint calibration updates, and report results on all seven benchmarks with complete coverage over three seeds under the main 1024-token protocol.

The complete decoder accelerates generation across the full benchmark suite, with mean wall-clock speedups ranging from $1.59\times$ to $1.70\times$. Throughput improvements closely track wall-clock speedup, indicating that the observed gains reflect faster generation rather than differences in output length alone. Mean final-answer accuracy is higher on MATH-500, AMC 2023, and Minerva/OCW, unchanged on AIME 2025 and AIME 2026, and lower on AIME 2024 and OlympiadBench. Given the small AIME evaluation sets and seed-level variation, we report task accuracy separately from decoding speed rather than interpreting these differences as systematic quality gains or losses.

4.3 Ablation of Proposal Sources

We next disentangle the contributions of recurrent latent drafting and contextual reuse. *Latent-only* uses the recurrent drafter as the sole proposal source, *Lookup-only* uses contextual reuse without neural drafting, and *Hybrid* combines the two. Table 3 compares the three routes using the same Qwen3-8B models after Extended calibration and the same decoding protocol as Table 2.

The ablation reveals two complementary sources of acceleration. Latent drafting alone achieves a consistent net speedup across all seven benchmarks, with wall-clock gains of $1.20\times$–$1.27\times$, showing that the recurrent transition is efficient enough to compensate for its own recurrent drafting cost. Contextual reuse provides a stronger proposal path when suitable continuations are available because it avoids neural drafting altogether. Combining the two yields the highest speedup on every benchmark, indicating that the recurrent transition contributes useful speculative proposals beyond the contexts covered by lookup.

Table 4: Effect of recurrent joint calibration on Qwen3-8B. Each cell reports the mean over three seeds at Staged, Joint calibration, and Extended calibration. The color intensity is independently normalized within each metric.

	(a) Latent acceptance (%)			(b) Latent-only speedup			(c) Hybrid speedup		
Benchmark	**Staged**	**Joint calibration**	**Extended calibration**	**Staged**	**Joint calibration**	**Extended calibration**	**Staged**	**Joint calibration**	**Extended calibration**
MATH-500	19.44	24.66	26.68	1.16	1.24	1.27	1.56	1.61	1.63
AIME 2024	16.45	20.54	22.17	1.12	1.18	1.21	1.63	1.70	1.70
AIME 2025	16.43	20.59	21.95	1.11	1.18	1.20	1.61	1.66	1.65
AIME 2026	17.12	21.02	22.40	1.13	1.20	1.22	1.58	1.63	1.64
AMC 2023	19.40	23.79	25.29	1.18	1.25	1.27	1.61	1.67	1.67
Minerva/OCW	18.08	21.20	22.62	1.13	1.18	1.20	1.54	1.57	1.59
OlympiadBench	17.83	22.07	23.49	1.14	1.21	1.23	1.61	1.65	1.66

4.4 Effect of Recurrent Joint Calibration

We next study whether recurrent joint calibration improves the drafter beyond staged initialization. For each seed, the Joint calibration and Extended calibration models are obtained by continuing from the same Staged model, with the target kept fixed and only T_ϕ and B_ψ updated. This allows us to attribute changes in drafting behavior directly to recurrent calibration. Table 4 reports acceptance and wall-clock speedup for latent drafting alone, together with the resulting Hybrid speedup. Extended calibration uses a larger training budget and is included to examine whether the gains persist with continued calibration.

Recurrent calibration produces a consistent improvement in the neural drafter. From Staged to Joint calibration, both draft acceptance and wall-clock speedup for latent drafting alone increase across the entire benchmark suite; Extended calibration further improves both quantities on every task. The effect is therefore visible not only in proposal quality but also in decoding efficiency. Hybrid speedup follows the same overall trend, although the gains become less monotonic once lookup is included because total latency also depends on the availability and length of contextual matches.

The same behavior appears at Qwen3-4B scale. After 4,000 joint-calibration steps, Latent-only decoding moves from approximately break-even to a consistent net speedup across all seven benchmarks. Complete Qwen3-4B results and the equal-budget full-joint comparison are provided in Appendix C.1.

4.5 Inference Sensitivity

We finally examine how inference choices affect the realized speedup. We vary the verification threshold λ and the position at which speculative decoding begins, while keeping the trained model fixed. We also compare the default response budget of 1024 tokens with a longer budget of 4096 tokens on the subset of benchmarks with complete coverage. Table 5 summarizes both effects.

The verification threshold has only a modest effect that varies across tasks on wall-clock performance: relaxing λ from 1.00 to 0.90 produces small gains on some benchmarks but does not consistently dominate. In contrast, the speculation start position has a clear effect. Starting at token 32 is faster than delaying speculation to token 128 on every benchmark, since a larger fraction of the generated response can benefit from speculative decoding.

Longer responses also provide more opportunity to amortize drafting and verification overhead. On the four benchmarks with complete 4096-token coverage under Extended calibration, Hybrid speedup increases from $1.64\times$–$1.70\times$ at a budget of 1024 tokens to $2.17\times$–$2.53\times$ with a budget of 4096 tokens. We therefore treat response length as an important factor in realized acceleration, while leaving the full 4096-token breakdown to Appendix C.5.

Table 5: Inference sensitivity of Qwen3-8B after Extended calibration. Panel (a) reports Hybrid wall-clock speedup under different verification thresholds and speculation start positions. Panel (b) compares Hybrid speedup under 1024- and 4096-token response budgets for benchmarks with complete coverage over three seeds.

	(a) Verification threshold and speculation start			
Dataset	Start 32, λ=1.00	Start 32, λ=0.90	Start 128, λ=1.00	Start 128, λ=0.90
MATH-500	1.63	1.64	1.51	1.52
AIME 2024	1.70	1.70	1.64	1.65
AIME 2025	1.65	1.68	1.60	1.60
AIME 2026	1.64	1.66	1.55	1.58
AMC 2023	1.67	1.69	1.57	1.59
Minerva/OCW	1.59	1.58	1.50	1.50
OlympiadBench	1.66	1.68	1.58	1.59

	(b) Response-length sensitivity	
Dataset	budget of 1024 tokens	budget of 4096 tokens
	Hybrid speedup	Hybrid speedup
AIME 2024	1.70	2.53
AIME 2025	1.65	2.23
AIME 2026	1.64	2.17
AMC 2023	1.67	2.25

5 Discussion and Limitations

The ablations show that the main components of OneiroDraft contribute complementary gains: recurrent latent drafting provides net acceleration on its own, joint calibration improves both draft acceptance and Latent-only speed, and contextual reuse provides an additional inexpensive proposal path. The remaining limitations largely reflect the general trade-offs of speculative decoding. Longer draft horizons can reduce the number of target-model decoding steps, but later proposals are harder to predict, while increasing drafter capacity can improve proposal quality only at additional drafting cost. OneiroDraft addresses this balance by repurposing a small interval of target Transformer layers as a shared recurrent transition, calibrating it on predicted-state trajectories, and re-anchoring to states computed by the target model after each verification round. The current implementation uses a fixed boundary, transition interval, and rollout depth, leaving room to adapt these choices to the predictability of the current trajectory. Contextual reuse is likewise available only when a matching continuation exists, while recurrent latent drafting provides a complementary proposal path when lookup is unavailable. The present evaluation focuses on Qwen3 mathematical reasoning, and extending the same target-layer repurposing principle to additional model families and generation domains will further test its broader applicability.

6 Conclusion

We introduced OneiroDraft, which repurposes a contiguous interval of the pretrained target Transformer from representation transformation along model depth into a shared cross-position transition for speculative drafting. A staged-to-joint training procedure improves recurrent state prediction and token readout, while contextual reuse provides an additional inexpensive proposal path. Across seven mathematical reasoning benchmarks with Qwen3-8B evaluation, recurrent latent drafting alone achieves $1.20\times$–$1.27\times$ wall-clock speedup, and the complete Hybrid decoder reaches $1.59\times$–$1.70\times$ over paired greedy decoding while largely preserving task accuracy. These results show that, on the evaluated Qwen3 mathematical reasoning setting, pretrained target computation can be adapted into a low-cost recurrent mechanism for speculative generation.

AI USE STATEMENT

Generative AI tools assisted code review, experiment bookkeeping, plotting, and language editing. They did not generate benchmark labels or replace any reported run. The authors checked all numerical claims against frozen result files and take responsibility for the submission.

ETHICS STATEMENT

This work uses public models and benchmarks and collects no human-subject data. Faster decoding inherits the target model's capabilities and failures; deployment should retain its safety controls.

REPRODUCIBILITY STATEMENT

The supplement includes code, frozen tables, hashes, schedules, complete results, and rejected controls. Every figure is regenerated from these files.

REFERENCES

Zachary Ankner, Rishab Parthasarathy, Aniruddha Nrusimha, Christopher Rinard, Jonathan Ragan-Kelley, and William Brandon. Hydra: Sequentially-dependent draft heads for medusa decoding. *arXiv preprint arXiv:2402.05109*, 2024.

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

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D Lee, Deming Chen, and Tri Dao. Medusa: Simple llm inference acceleration framework with multiple decoding heads. *arXiv preprint arXiv:2401.10774*, 2024.

Yuxuan Cai, Xiaozhuan Liang, Xinghua Wang, Jin Ma, Haijin Liang, Jinwen Luo, Xinyu Zuo, Lisheng Duan, Yuyang Yin, and Xi Chen. Fastmtp: Accelerating llm inference with enhanced multi-token prediction. *arXiv preprint arXiv:2509.18362*, 2025.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*, 2023.

Yuetao Chen, Xuliang Wang, Xinzhou Zheng, Ming Li, Peng Wang, and Hong Xu. Make every draft count: Hidden state based speculative decoding. *arXiv preprint arXiv:2602.21224*, 2026.

Yunfei Cheng, Aonan Zhang, Xuanyu Zhang, Chong Wang, and Yi Wang. Recurrent drafter for fast speculative decoding in large language models. *arXiv preprint arXiv:2403.09919*, 2024.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. *arXiv preprint arXiv:1807.03819*, 2018.

Doğaç Eldenk, Payal Mohapatra, Yigitcan Comlek, Kaan Oktay, Hongyang Zhang, and Stephen Xia. Attention drift: What autoregressive speculative decoding models learn. *arXiv preprint arXiv:2605.09992*, 2026.

Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, et al. Layerskip: Enabling early exit inference and self-speculative decoding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12622–12642, 2024.

Fabian Gloeckle, Badr Youbi Idrissi, Baptiste Rozière, David Lopez-Paz, and Gabriel Synnaeve. Better & faster large language models via multi-token prediction. *arXiv preprint arXiv:2404.19737*, 2024.

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, 2024a.

Zhenyu He, Zexuan Zhong, Tianle Cai, Jason Lee, and Di He. Rest: Retrieval-based speculative decoding. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 1582–1595, 2024b.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Yuxuan Hu, Ke Wang, Xiaokang Zhang, Fanjin Zhang, Cuiping Li, Hong Chen, and Jing Zhang. Sam decoding: Speculative decoding via suffix automaton. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12187–12204, 2025.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International conference on machine learning*, pp. 19274–19286. PMLR, 2023.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. Solving quantitative reasoning problems with language models. *Advances in neural information processing systems*, 35:3843–3857, 2022.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle: Speculative sampling requires rethinking feature uncertainty. *arXiv preprint arXiv:2401.15077*, 2024a.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle-2: Faster inference of language models with dynamic draft trees. In *Proceedings of the 2024 conference on empirical methods in natural language processing*, pp. 7421–7432, 2024b.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle-3: Scaling up inference acceleration of large language models via training-time test. *Advances in Neural Information Processing Systems*, 38:136737–136756, 2026.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024.

Tianyu Liu, Yuhao Shen, Xinyi Hu, Baolin Zhang, Hengxin Zhang, Jun Dai, Jun Zhang, Shuang Ge, Lei Chen, Yue Li, et al. When hidden states drift: Can kv caches rescue long-range speculative decoding? *arXiv preprint arXiv:2604.26412*, 2026.

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

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Chang Shi, Tim Pearce, Manan Tomar, Siddhartha Sen, and John Langford. Hierarchical latent prediction for language models. *arXiv preprint arXiv:2608.05806*, 2026.

Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. Blockwise parallel decoding for deep autoregressive models. *Advances in Neural Information Processing Systems*, 31, 2018.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pp. 38–45, 2020.

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.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. Metamath: Bootstrap your own mathematical questions for large language models. In *International Conference on Learning Representations*, volume 2024, pp. 45040–45061, 2024.

Jun Zhang, Jue Wang, Huan Li, Lidan Shou, Ke Chen, Gang Chen, and Sharad Mehrotra. Draft& verify: Lossless large language model acceleration via self-speculative decoding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11263–11282, 2024.

Lefan Zhang, Xiaodan Wang, Yanhua Huang, and Ruiwen Xu. Learning harmonized representations for speculative sampling. In *International Conference on Learning Representations*, volume 2025, pp. 35367–35388, 2025.

Yongchao Zhou, Kaifeng Lyu, Ankit Singh Rawat, Aditya Krishna Menon, Afshin Rostamizadeh, Sanjiv Kumar, Jean-François Kagy, and Rishabh Agarwal. Distillspec: Improving speculative decoding via knowledge distillation. In *International Conference on Learning Representations*, volume 2024, pp. 32011–32050, 2024.

A Additional Analysis of Target-Derived Recurrent Drafting

This section provides additional analysis of the target-derived recurrent drafting formulation. We first characterize when intermediate target-model representations admit a predictable cross-position evolution, then explain why a contiguous interval of pretrained target layers provides a useful starting point for the recurrent transition. We finally analyze error accumulation, target re-anchoring, token-level agreement, and the conditions under which the repurposed transition yields net decoding acceleration. Unless otherwise stated, the analysis assumes strict verification, which is the default setting in the main experiments.

A.1 Why Can Target Hidden States Be Recurrently Predicted?

Let

$$h_t = H_\theta(x_{\le t}) \tag{13}$$

denote the target representation at the selected boundary after processing prefix $x_{\le t}$. Under greedy decoding, the next token is deterministically selected as

$$x_{t+1} = \arg\max_v p_\theta(v \mid x_{\le t}), \tag{14}$$

and the next boundary state h_{t+1} is consequently determined by the extended prefix $x_{\le t+1}$.

This observation alone does not imply that h_t is a strict Markov state. In particular, an intermediate representation can in principle discard information from the prefix that becomes relevant at later positions. The more appropriate question is therefore whether the target representation retains sufficient information for its *next latent state to be predictable with small residual uncertainty*.

Let

$$Z_t = (h_1, \dots, h_t) \tag{15}$$

denote the target hidden-state history available to the recurrent transition. Consider the population one-step prediction objective

$$\mathcal{R}(T) = \mathbb{E}\left[\|T(Z_t) - h_{t+1}\|_2^2\right]. \tag{16}$$

Proposition 1: Predictability of the latent transition. Among all deterministic predictors, the minimum-MSE transition is

$$T^\star(Z_t) = \mathbb{E}\left[h_{t+1} \mid Z_t\right], \tag{17}$$

and its irreducible prediction error is

$$\mathcal{R}(T^\star) = \mathbb{E}\left[\operatorname{tr}\operatorname{Cov}\left(h_{t+1} \mid Z_t\right)\right]. \tag{18}$$

Proof. This follows directly from the standard conditional-expectation decomposition of squared error:

$$\begin{aligned}\mathbb{E}\left[\|T(Z_t) - h_{t+1}\|_2^2\right] &= \mathbb{E}\left[\|T(Z_t) - \mathbb{E}[h_{t+1} \mid Z_t]\|_2^2\right] \\ &\quad + \mathbb{E}\left[\operatorname{tr}\operatorname{Cov}(h_{t+1} \mid Z_t)\right].\end{aligned} \tag{19}$$

The first term is minimized by $T(Z_t) = \mathbb{E}[h_{t+1} \mid Z_t]$. □

Proposition 1 gives a precise interpretation of recurrent latent drafting. Predicting future target states does not require the selected representation to be an exact Markov state. Instead, it requires the conditional uncertainty in Eq. 18 to be sufficiently small on the trajectory distribution relevant to generation. If two substantially different prefixes collapse to indistinguishable latent histories while inducing different future target states, this uncertainty becomes irreducible. Conversely, when the boundary representation preserves the prefix information relevant to subsequent target computation, the next hidden state can be well approximated by a deterministic transition.

There are several reasons to expect an intermediate target representation to satisfy this property locally. First, each h_t has already undergone substantial target-model computation and is conditioned through causal self-attention on the preceding context. Second, the representation is learned precisely

to support subsequent target layers and next-token prediction, so it retains information useful for continuation. Third, OneiroDraft predicts within the target representation space rather than constructing an independent latent space whose semantics must be learned from scratch.

This perspective is related to prior feature-based speculative methods. EAGLE demonstrates that autoregressive prediction of target-model features can provide effective speculative proposals, while ReDrafter conditions a recurrent drafter directly on target hidden states (Li et al., 2024a; Cheng et al., 2024). At the same time, feature prediction should not be interpreted as universally easier or sufficient by itself: EAGLE-3 reports limitations of explicit feature reconstruction and instead emphasizes direct token prediction with multi-layer feature fusion (Li et al., 2026). Our design therefore does not rely on latent reconstruction alone. The recurrent trajectory is constrained in hidden space while the boundary readout is simultaneously trained to match the target's token distribution.

A.2 Repurposing Target Layers as a Cross-Position Transition

The transition T_ϕ is initialized from a small contiguous stack of Transformer layers copied from the target model. A target block transforms representations along model depth rather than across token positions. Repurposing it as a cross-position transition therefore changes its computational role: training establishes the new dynamics, while the copied parameters provide a pretrained sequence-processing initialization in the same hidden-state space.

This design is motivated by three considerations.

Initialization in the target representation space. Adjacent target states can already be geometrically close, so the purpose of copying is not to guarantee a smaller one-step displacement than an identity map. Instead, compared with random base weights, copied layers bring normalization statistics, attention projections, feed-forward transformations, residual scales, and positional conventions learned on the target model's own hidden activations. Fine-tuning can then adapt this pretrained transformation to the cross-position prediction objective.

This adaptation can be viewed schematically as

$$T_\phi = T_{\phi_0} + \Delta_\phi, \tag{20}$$

where T_{ϕ_0} denotes the copied initialization and Δ_ϕ represents the functional change induced by transition training. Equation 20 is conceptual rather than an architectural decomposition.

Causal sequence-processing prior. Predicting $\hat{h}_{t+k}$ requires conditioning on both the verified hidden-state prefix and the previously predicted latent states. A Transformer block already implements causal sequence processing with self-attention and KV caching, matching the dependency structure required by Eq. 2. The copied block therefore supplies an appropriate computational structure for recurrent rollout in addition to a pretrained parameter initialization.

Preserving the representation coordinate system. The predicted states are consumed by a lightweight readout that reuses components of the target model. Large departures from the scale and organization of the selected hidden-state space would make this reuse harder. Hidden-state supervision during training therefore complements the copied initialization by keeping the recurrent trajectory in the representation space expected by the readout.

These considerations motivate the architecture; the diagnostic below directly tests only the initialization comparison between copied and randomly resampled transition weights.

This design is conceptually related to self-speculative methods that reuse computation from the target itself. Draft & Verify accelerates drafting by selectively skipping target layers, while LayerSkip trains intermediate layers to support early-exit predictions and uses the remaining target layers for verification (Zhang et al., 2024; Elhoushi et al., 2024). OneiroDraft differs in that the copied layers are adapted into a recurrent operator whose role is to extrapolate the hidden trajectory across positions rather than to execute an unchanged subset of the target network.

Table 6: One-step initialization diagnostic on Llama-3-8B-Instruct before additional transition training. Lower is better for MSE, relative MSE, and teacher KL; higher is better for cosine similarity and top-1 agreement.

Initialization	Hidden MSE ↓	Rel. MSE ↓	Cosine ↑	Teacher top-1 (%) ↑	Label top-1 (%) ↑	Teacher KL ↓
Identity sanity baseline	0.553	3.015	0.542	–	–	–
Target-derived	0.654	3.602	0.436	59.70	44.05	6.986
Random init.	15.259	88.843	0.064	0.02	0.02	15.058

A.3 Initialization Compatibility Probe

We next compare the copied initialization with two one-step baselines on Llama-3-8B-Instruct. The recorded configuration predicts h_{t+1} from h_t, uses anchor layer 28 and copied target layers 29–30, and evaluates 64 MATH-500 examples comprising 6,054 valid tokens in BF16 with training seed 42. *Target-derived* copies the target layers with a fresh zero-delta transition LoRA; *Random init.* keeps the same transition architecture and parameter count but resamples only the transition base weights from a normal initializer with standard deviation 0.02; and *Identity* directly uses h_t as the prediction of h_{t+1}. The target model, adapters outside the transition base weights, readout, tokenizer, and inputs are fixed across the controlled Target-derived/Random comparison. Table 6 reports the resulting one-step metrics.

The Identity baseline matches or exceeds the copied transition on all three geometric metrics: it has lower hidden MSE (0.553 vs. 0.654), lower relative MSE (3.015 vs. 3.602), and higher cosine similarity (0.542 vs. 0.436). This indicates that adjacent target states are already locally close and that one-step geometric proximity alone does not distinguish the identity map from the copied transition.

The controlled Target-derived/Random comparison tests a different question. With architecture, readout, target model, tokenizer, and inputs held fixed, copying the pretrained transition weights reduces hidden MSE from 15.259 to 0.654, increases cosine similarity from 0.064 to 0.436, raises teacher top-1 agreement from 0.02% to 59.70%, and lowers teacher KL from 15.058 to 6.986. Token-level metrics were not computed for the Identity sanity baseline in this probe. The result therefore supports copying as a substantially stronger initialization than random base weights, while not implying that it minimizes one-step geometric error.

A.4 Error Accumulation and the Role of Re-Anchoring

A central difficulty of recurrent drafting is that prediction errors can feed back into subsequent transition steps. To obtain a local stability bound, we treat the available prefix and latent history as part of the effective recurrent state and suppress this context from the notation. Along a realized trajectory, we write the true latent evolution as

$$h_{i+1} = \mathcal{G}(h_i), \tag{21}$$

and the learned evolution as

$$\hat{h}_{i+1} = T_\phi(\hat{h}_i). \tag{22}$$

This shorthand does not assume that an individual boundary vector is Markov; the omitted history is part of the effective state on which the transition is conditioned.

Assume that, within a neighborhood of the target trajectory,

$$\|T_\phi(h) - \mathcal{G}(h)\|_2 \le \epsilon \tag{23}$$

and that T_ϕ is locally L-Lipschitz:

$$\|T_\phi(u) - T_\phi(v)\|_2 \le L\|u - v\|_2. \tag{24}$$

Let

$$e_k = \|\hat{h}_{t+k} - h_{t+k}\|_2. \tag{25}$$

Then

$$\begin{aligned} e_{k+1} &= \left\|T_\phi(\hat{h}_{t+k}) - \mathcal{G}(h_{t+k})\right\|_2 \\ &\le \left\|T_\phi(\hat{h}_{t+k}) - T_\phi(h_{t+k})\right\|_2 + \|T_\phi(h_{t+k}) - \mathcal{G}(h_{t+k})\|_2 \\ &\le Le_k + \epsilon. \end{aligned} \tag{26}$$

Starting from a target-computed state, $e_0 = 0$, and therefore

$$e_k \le \epsilon \sum_{j=0}^{k-1} L^j = \begin{cases} \epsilon \dfrac{L^k - 1}{L - 1}, & L \neq 1, \\ k\epsilon, & L = 1. \end{cases} \tag{27}$$

This simple bound captures two properties of recurrent latent drafting. First, longer speculative trajectories are intrinsically more difficult because even small one-step errors can accumulate through recurrent feedback. This motivates training T_ϕ on its own predicted-state distribution rather than only on teacher-forced one-step transitions. The non-detached joint rollout used by OneiroDraft directly optimizes this recursive computation.

Second, target verification provides more than token correctness: it periodically *resets the latent error*. After a speculative block is verified, temporary predicted states are discarded and the next neural rollout is initialized from target-computed hidden states. Consequently, the error in Eq. 27 accumulates only within an individual speculative block rather than across the entire generated sequence. If the maximum neural rollout depth is K, the relevant worst-case bound is e_K regardless of the total response length. This re-anchoring property is particularly important for long-form generation, where an unrestricted recurrent latent model would otherwise have to remain stable over hundreds or thousands of consecutive prediction steps.

The analysis also explains why increasing rollout depth does not necessarily improve decoding efficiency. Larger K increases the maximum number of tokens that can be accepted per verification pass, but simultaneously increases latent prediction error and can reduce later-position acceptance. The optimal rollout length therefore depends on the balance between these two effects.

A.5 From Latent Prediction Error to Token Agreement

Hidden-state reconstruction is useful only insofar as the resulting states produce candidate tokens consistent with the target model's decisions. We next make this connection explicit.

Let

$$z_i^\theta \in \mathbb{R}^{|\mathcal{V}|} \tag{28}$$

denote the target-model logits for the next token at position i, and let

$$\tilde{z}_i = B_\psi(\hat{h}_i) \tag{29}$$

denote the draft logits decoded from a predicted hidden state.

Suppose the readout is calibrated on the corresponding true target state such that

$$\left\| B_\psi(h_i) - z_i^\theta \right\|_\infty \le \delta_i, \tag{30}$$

and is locally Lipschitz around h_i:

$$\left\| B_\psi(\hat{h}_i) - B_\psi(h_i) \right\|_\infty \le L_B \|\hat{h}_i - h_i\|_2. \tag{31}$$

Then

$$\|\tilde{z}_i - z_i^\theta\|_\infty \le \delta_i + L_B e_i. \tag{32}$$

Let y_i denote the target top-1 token and define its logit margin as

$$\mu_i = z_{i,y_i}^\theta - \max_{v \neq y_i} z_{i,v}^\theta. \tag{33}$$

Proposition 2: Sufficient condition for strict acceptance. If

$$\delta_i + L_B e_i < \frac{\mu_i}{2}, \tag{34}$$

then

$$\arg\max_v B_\psi(\hat{h}_i)_v = \arg\max_v z_{i,v}^\theta, \tag{35}$$

and the corresponding candidate satisfies strict top-1 verification.

Proof. Every draft logit differs from its target counterpart by at most $\delta_i + L_B e_i$. Therefore, the difference between the target top-1 logit and any competing logit can decrease by at most $2(\delta_i + L_B e_i)$. Condition 34 guarantees that the original positive margin remains positive. □

Proposition 2 clarifies why the two components of the joint objective in Sec. 3.3 are complementary. The hidden-state objective reduces e_i, keeping recurrent predictions close to the target latent trajectory. Distribution distillation directly improves the behavior of the readout and reduces the discrepancy represented by δ_i. Optimizing only state reconstruction is insufficient because even a small latent error may change the top-ranked token when the target margin μ_i is small. Conversely, matching token distributions without controlling the latent trajectory can allow recurrent states to drift away from the region on which the readout was trained. Joint recurrent calibration addresses both sources of error.

The same analysis also shows why perfect hidden-state reconstruction is unnecessary. Strict acceptance only requires the predicted representation to remain within a decision-preserving neighborhood whose size depends on the target token margin. States may therefore differ substantially in directions that have little influence on the vocabulary projection while still producing the same accepted token.

A.6 A Simple Efficiency Condition for Recurrent Drafting

Finally, we relate draft quality to end-to-end decoding efficiency. Consider a speculative round with at most K neural draft tokens. Let

$$\xi_j = \Pr\left(d_{t,j} \text{ is accepted} \mid d_{t,1:j-1} \text{ are accepted}\right) \tag{36}$$

be the conditional acceptance probability of the j-th candidate. Because verification commits only the longest accepted prefix, the expected number of accepted draft tokens is

$$\mathbb{E}[A_K] = \sum_{j=1}^{K} \prod_{r=1}^{j} \xi_r. \tag{37}$$

Including the target-generated anchor, the expected progress of one speculative round is therefore

$$G_K = 1 + \sum_{j=1}^{K} \prod_{r=1}^{j} \xi_r. \tag{38}$$

Let C_{AR} denote the average cost of one standard autoregressive target step, $C_{\mathrm{D}}(K)$ the cost of producing K neural candidates, and $C_{\mathrm{V}}(K+1)$ the cost of verifying the anchor and candidates in one target forward pass. Ignoring secondary implementation effects, the approximate speedup is

$$S(K) \approx \frac{G_K C_{\mathrm{AR}}}{C_{\mathrm{D}}(K) + C_{\mathrm{V}}(K+1)}. \tag{39}$$

Recurrent drafting therefore provides net acceleration whenever

$$C_{\mathrm{D}}(K) + C_{\mathrm{V}}(K+1) < G_K C_{\mathrm{AR}}. \tag{40}$$

Equation 40 makes explicit the central design objective of OneiroDraft. Increasing transition capacity can improve the acceptance probabilities ξ_j, but only if the resulting increase in G_K compensates for the additional drafting cost. Conversely, an extremely inexpensive transition is not useful if recurrent error causes the later ξ_j values to collapse. Reusing a small target-derived transition is intended to operate between these extremes: it provides a proposal mechanism derived from the target model while remaining substantially cheaper than executing the complete target model.

The same view explains the hybrid proposal strategy. Contextual reuse has very low proposal cost when a matching continuation is available, whereas recurrent latent drafting provides broader coverage when lookup fails. The hybrid system therefore reduces the expected proposal cost on lookup hits without restricting speculative decoding to contexts for which reusable continuations already exist.

Taken together, these analyses characterize recurrent latent drafting as a short-horizon approximation to the target model's latent dynamics. Its effectiveness relies neither on the selected hidden state being

Table 7: Recurrent layout and adapter dimensions for the Qwen3 configurations. Layer indices are zero-based. Target LoRA rank 8 is used on both sides of the recurrent boundary.

Configuration	Qwen3-4B	Qwen3-8B
Hidden dimension d	2560	4096
Recurrent boundary	`hidden_states[33]`	`hidden_states[33]`
Copied interval $\mathcal{I}$	layers 33–34	layers 33–34
Target LoRA rank	8	8
Transition LoRA rank	80	128
Boundary adapter rank	160	256

perfectly Markov nor on the copied target layers already implementing the desired temporal transition. Instead, the method requires that future target states be locally predictable from the available target representation, that the learned transition remain sufficiently stable over short recurrent rollouts, and that the resulting latent error remain within a region that preserves the target's token decisions. Target-derived initialization, rollout-aware joint training, and periodic re-anchoring each address a different part of these requirements.

B Training and Evaluation Details

B.1 Training Configuration

Data and models. We train the drafting modules on MetaMathQA. The paper-facing benchmark study uses Qwen3-4B and Qwen3-8B, with seeds 42, 43, and 44 for the replicated experiments. Both targets contain 36 decoder layers. Backbone weights remain frozen; trainable changes to the target or recurrent transition are implemented with LoRA. Llama-3-8B-Instruct is used only for the supplementary optimization trace and initialization diagnostic in Appendices B.2 and A.3.

Recurrent boundary and copied layers. For both Qwen3 models, the recurrent representation is

$$h_t = \texttt{hidden_states[33]},$$

i.e., the output of zero-based decoder layer 32 and the input to layer 33. The recurrent transition is initialized by independently copying decoder layers 33–34 inclusive, corresponding to `layers[33:35]`. The target model still executes all 36 layers during verification; the copied interval is a separate recurrent module rather than an early-exit path. Table 7 summarizes the model-specific dimensions and adapter ranks.

The target LoRA uses scaling parameter $\alpha_{\text{LoRA}} = 16$ with rank 8. The transition LoRA uses rank/scaling pairs $(80, 160)$ and $(128, 256)$ for Qwen3-4B and Qwen3-8B, respectively, so $\alpha_{\text{LoRA}}/r = 2$ in all three cases. LoRA is applied to the attention projections `q_proj`, `k_proj`, `v_proj`, and `o_proj`, and to the feed-forward projections `gate_proj`, `up_proj`, and `down_proj`. The symbol α_{LoRA} is unrelated to the loss weight α in Eq. 8.

Boundary readout parameterization. The boundary readout applies a low-rank residual adapter before reusing the target model's final normalization and vocabulary projection:

$$\widetilde{h} = h + U_\psi V_\psi N_\psi(h), \tag{41}$$

$$q_\psi(v \mid h) = \text{softmax}\left(W_{\text{LM}} N_{\text{f}}(\widetilde{h})\right)_v. \tag{42}$$

Here N_ψ is trainable, $V_\psi : \mathbb{R}^d \to \mathbb{R}^{r_b}$ and $U_\psi : \mathbb{R}^{r_b} \to \mathbb{R}^d$ are bias-free linear maps, and r_b is 160 for Qwen3-4B and 256 for Qwen3-8B. The adapter contains no intermediate activation and no LoRA-style scaling factor. V_ψ uses Kaiming initialization and U_ψ is zero-initialized. N_{f} and W_{LM} are reused from the target model. This boundary adapter is enabled during readout initialization and joint calibration; the earlier transition stages use no boundary adapter.

Optimization. We use AdamW (Loshchilov & Hutter, 2017) with zero weight decay, a maximum gradient norm of 1, and a constant learning-rate schedule with no learning-rate warmup. Training uses BF16 precision, an effective batch size of 8 sequences per optimization step, and a sequence cutoff of 1024 tokens.

Staged transition training. The replicated Qwen training chain first performs 49,199 core transition updates at 1×10^{-5} and then 9,840 detached multistep updates at 3×10^{-6}. During these stages, the target LoRA and transition LoRA are trainable. The readout is subsequently trained for 222 updates at 1×10^{-4} and 762 updates at 5×10^{-5}, giving a 60,023-update staged checkpoint for the replicated branches.

During the detached multistep stage, predicted states are reused while gradients through the reused trajectory are blocked:

$$\hat{h}^{\mathrm{det}}_{t+k} = \tau_\phi\left(h_{\le t} \oplus \mathrm{sg}\left(\hat{h}^{\mathrm{det}}_{t+1:t+k-1}\right)\right), \qquad k = 1, \dots, K. \tag{43}$$

The staged multistep configuration uses $K = 4$, a context window of 128 latent positions, and geometric weighting 0.7^{k-2} for the hidden-state terms from $k = 2$ onward, normalized by the sum of valid weights. It also includes a delta-cosine auxiliary term with coefficient 0.02. Thus, this detached stage is not the same objective as the uniformly averaged joint recurrence in Eq. 6; its role is to expose the transition to its own imperfect predicted states without propagating later losses through earlier predictions.

Readout initialization. For a target-computed boundary state h_t, the readout is initialized with next-token supervision and distribution distillation:

$$\mathcal{L}_{\mathrm{readout}} = \mathbb{E}_{x,t}\left[-\eta_{\mathrm{CE}} \log q_\psi(x_{t+1} \mid h_t) + \eta_{\mathrm{KL}} D_{\mathrm{KL}}\left(\bar{p}_t \,\|\, q_\psi(\cdot \mid h_t)\right)\right], \tag{44}$$

with

$$\eta_{\mathrm{CE}} = 1.0, \qquad \eta_{\mathrm{KL}} = 0.2.$$

Training positions are sampled with stride 8 and at most 64 positions per sequence. In the reported Qwen branches, $\bar{p}_t$ is the next-token distribution of the adapted target, which is held fixed during readout initialization. During joint calibration, the direct readout CE/KL terms are disabled and distribution supervision is supplied by the recurrent rollout loss.

Smooth-L_1 state distance. The joint recurrent state objective uses the element-wise smooth-L_1 penalty

$$\rho_\beta(r) = \begin{cases} \dfrac{r^2}{2\beta}, & |r| \le \beta, \\ |r| - \dfrac{\beta}{2}, & |r| > \beta, \end{cases} \qquad \beta = 0.1, \tag{45}$$

with

$$\ell_\beta(u, v) = \frac{1}{d} \sum_{j=1}^{d} \rho_\beta(u_j - v_j). \tag{46}$$

This is the ℓ_β used in Eq. 6. The first predicted state is handled by $\mathcal{L}_{\mathrm{step}}$, so the additional recurrent state term begins at $k = 2$.

Joint calibration weights and schedule. The recurrent objective in Eq. 8 uses

$$\alpha = 0.1, \qquad \gamma = 10, \qquad K = 2,$$

and a linearly warmed recurrent weight

$$w(s) = 0.1 \min\left(1, \frac{s+1}{200}\right), \qquad s = 0, 1, 2, \dots, \tag{47}$$

where s is the number of completed optimizer updates within the joint stage. The first joint update therefore uses $w(0) = 5 \times 10^{-4}$, and the schedule reaches its maximum value 0.1 at the 200th update. After this warmup, the recurrent backbone of the objective is

$$0.1\,\mathcal{L}_{\mathrm{step}} + 0.1\,\mathcal{L}_{\mathrm{state}} + 1.0\,\mathcal{L}_{\mathrm{dist}}. \tag{48}$$

The distribution loss uses temperature 1 and the direction $D_{\mathrm{KL}}(p_\theta \| q_\psi)$ with detached teacher logits. The joint rollout uses the full available prefix, no rollout detachment, no scheduled sampling, and no within-rollout decay. Up to four rollout starts are selected at stride 64; Qwen3-4B uses linearly spaced starts and Qwen3-8B uses cyclic start selection.

Table 8: Training schedule for the replicated Qwen staged-to-joint branches. For Qwen3-8B, Extended calibration continues the replicated 60,023-update Staged checkpoint for 20,000 joint-calibration updates, reaching 80,023 cumulative updates.

Stage	Updates	Learning rate	Trainable component
Core transition	49,199	1×10^{-5}	target LoRA + transition LoRA
Detached multistep	9,840	3×10^{-6}	target LoRA + transition LoRA
Readout warmup	222	1×10^{-4}	B_ψ
Readout continuation	762	5×10^{-5}	B_ψ
Joint calibration	4,000	3×10^{-6}	transition LoRA + B_ψ
Full-joint ablation	64,023 total	3×10^{-6}	target LoRA + transition LoRA + B_ψ

Variant-specific objective terms. Equation 8 is the recurrent backbone of the joint objective and directly matches the frozen-target Qwen3-8B post-joint configuration. The Qwen3-4B post-joint branch additionally uses

$$0.005\,\mathcal{L}_{\text{rel}} + 0.005\,\mathcal{L}_{\cos}, \tag{49}$$

where $\mathcal{L}_{\text{rel}}$ is the one-step relative-MSE term and $\mathcal{L}_{\cos} = 1 - \text{cosine}(\cdot, \cdot)$. Full-joint training also optimizes the target LoRA and adds the target/reference regularizer

$$\mathcal{R}_{\text{target}} = 20\,\mathcal{L}_{\text{final-MSE}} + \mathcal{L}_{\text{final-rel}} + \mathcal{L}_{\text{final-cos}} + 0.2\,\mathcal{L}_{\text{target-KL}}. \tag{50}$$

For Qwen3-4B full-joint training, Eq. 49 is present as well. The ordinary SFT cross-entropy weight is zero in these joint variants. These additions are variant-specific and are not part of the compact recurrent objective used to define the method.

Joint and extended training branches. Starting from the 60,023-update replicated staged checkpoint, the Qwen3-4B and Qwen3-8B joint branches train only the transition LoRA and boundary readout for 4,000 updates at 3×10^{-6}. The equal-budget Full-joint branches run for 64,023 total updates and also optimize the target LoRA. For Qwen3-8B, the Extended calibration branch continues the replicated 60,023-update Staged checkpoint for 20,000 joint-calibration updates, reaching 80,023 cumulative updates.

B.2 Training Dynamics

Figures 2–4 document the optimization trajectories used to monitor recurrent joint calibration. Qwen3-8B and Qwen3-4B correspond to the models used in the paper-facing benchmark study, while Llama-3-8B-Instruct is included only as a supplementary cross-backbone optimization trace. The plots report lightly smoothed three-seed means with shaded ± 1 sample standard deviation. They are included to characterize optimization behavior; the benchmark tables use the predefined training stages in Table 8 rather than selecting an evaluation point from the minimum observed loss.

B.3 Evaluation Protocol

Benchmarks. We evaluate MATH-500, AIME 2024, AIME 2025, AIME 2026, AMC 2023, Minerva/OCW, and OlympiadBench. The corresponding evaluation sets contain 500, 30, 30, 30, 40, 272, and 675 examples. We use the recorded task-specific answer extraction and scoring rules for each benchmark. Unless otherwise stated, generation is limited to 1024 new tokens with a maximum prompt length of 4096 tokens.

Decoding configuration. All paper-facing Qwen decoding experiments use BF16 precision and batch size 1. Neural drafting proposes at most two tokens per speculative round, while contextual lookup may propose up to eight; neither count includes the target-generated anchor. Unless explicitly varied, both proposal routes are enabled after 32 generated tokens have been committed and use strict verification with $\lambda = 1.00$. Before that point, decoding proceeds through the target model alone.

Contextual lookup. Lookup operates on exact token IDs and is local to each example. Let H denote the actual prompt token IDs followed by all generated tokens already committed for the

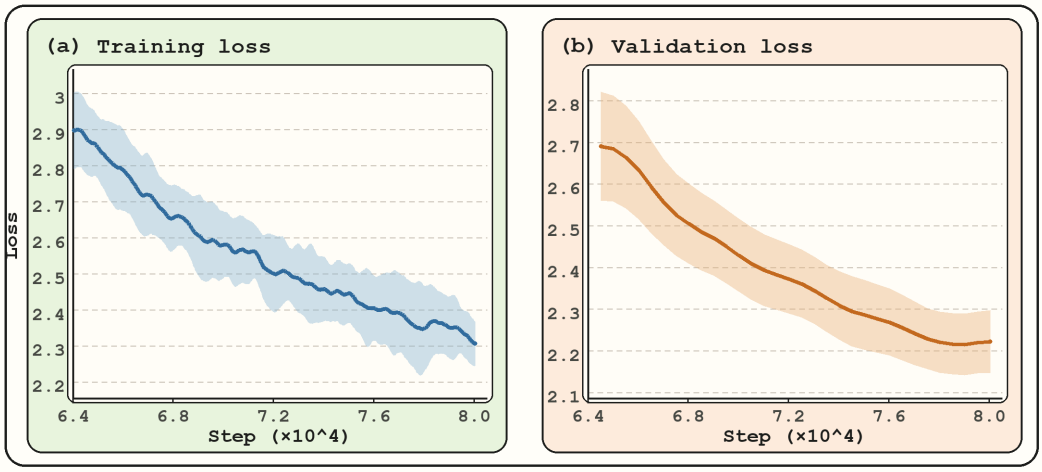

Figure 2: Training dynamics of ONEIRODRAFT on Qwen3-8B during extended recurrent joint calibration. (a) Training loss and (b) validation loss are plotted against cumulative optimization steps. Solid curves show the lightly smoothed mean over three seeds; shaded regions denote ± 1 sample standard deviation.

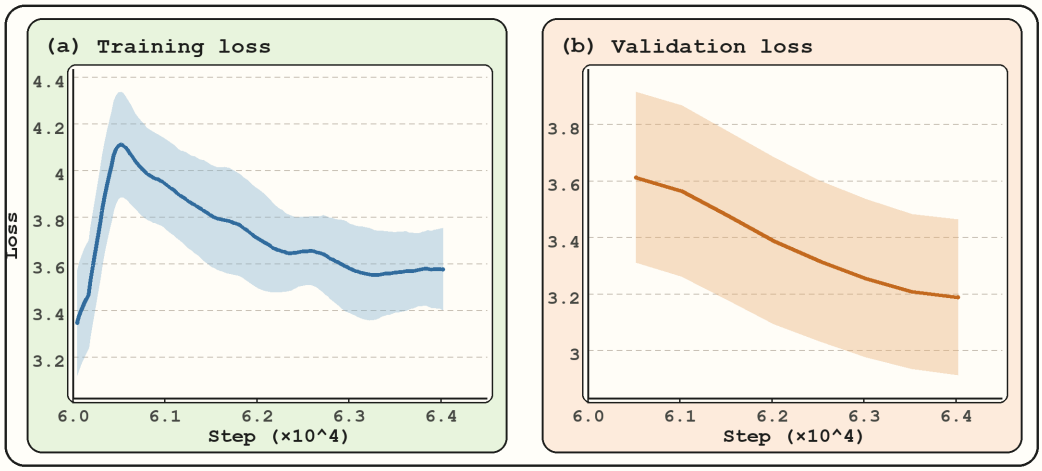

Figure 3: Training dynamics of ONEIRODRAFT on Qwen3-4B during recurrent joint calibration. (a) Training loss and (b) validation loss are plotted against cumulative optimization steps. Solid curves show the lightly smoothed mean over three seeds; shaded regions denote ± 1 sample standard deviation.

current example, and let a_t be the target-selected anchor for the current round. For $n = 5, 4, 3, 2$ in descending order, lookup forms

$$\kappa_n = H_{-(n-1):} \oplus (a_t) \tag{51}$$

and searches previous occurrences of this exact n-token key in H. For the first match length with a usable occurrence, the most recent occurrence is selected, provided that at least one token follows it in the committed history. Lookup then copies up to eight subsequent tokens, subject to the remaining generation budget. Longer exact matches take precedence over shorter ones even when the available continuation is shorter, and rejected speculative tokens are never inserted into the lookup history. The index includes the retained prompt and committed continuation, but it is never shared across examples.

Hybrid routing. After the speculation start position, Hybrid first queries contextual lookup. A nonempty lookup continuation is used as the complete draft for that round; neural drafting is not used to fill a shorter lookup proposal. When lookup returns no continuation, the recurrent transition

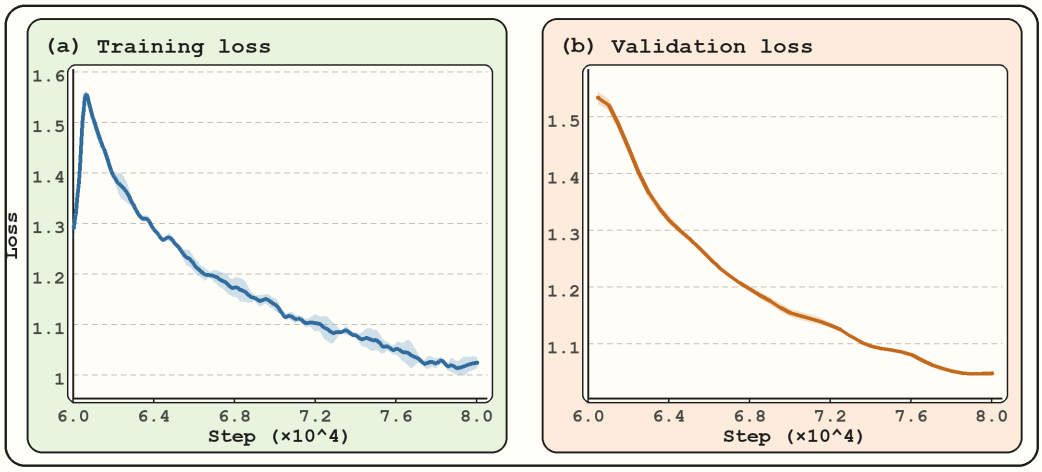

Figure 4: Supplementary training dynamics of ONEIRODRAFT on Llama-3-8B-Instruct during recurrent joint calibration. (a) Training loss and (b) validation loss are plotted against cumulative optimization steps. Solid curves show the lightly smoothed mean over three seeds; shaded regions denote ± 1 sample standard deviation. This trajectory is reported only as an optimization diagnostic and is not included in the primary benchmark tables.

proposes up to two neural draft tokens. After any rejection, routing is repeated from the newly committed prefix in the next round rather than switching proposal sources within the same verification round. Lookup-only decoding falls back to target-only decoding on a miss. Thus, contextual lookup and recurrent drafting are alternative proposal routes rather than components of one concatenated proposal.

Verification threshold. Using the generic candidate notation from Sec. 3.4, the target distribution for candidate $c_{t,j}$ is

$$\pi_{t,j}(v) = p_\theta\left(v \mid x_{\le t}, a_t, c_{t,<j}\right). \tag{52}$$

Its relative target support is

$$r_{t,j} = \frac{\pi_{t,j}(c_{t,j})}{\max_v \pi_{t,j}(v)}. \tag{53}$$

Strict verification at $\lambda = 1.00$ accepts a candidate only when its token ID equals the target argmax at that position. For $\lambda < 1$, the relaxed rule accepts when $r_{t,j} \ge \lambda$. In both cases, only the longest consecutively accepted prefix is committed. The lookup matching rule itself is independent of λ; the threshold applies only to target verification of the resulting candidates.

Paired baseline comparison. Every speculative run is paired with greedy decoding using the same target checkpoint. Wall-clock speedup is $T_{\text{greedy}}/T_{\text{ONEIRODRAFT}}$, and throughput speedup is $(N_{\text{ONEIRODRAFT}}/T_{\text{ONEIRODRAFT}})/(N_{\text{greedy}}/T_{\text{greedy}})$. For summaries over three seeds, each seed is first normalized by its own paired greedy run and the resulting ratios are then averaged over seeds 42, 43, and 44. The reported sample standard deviation therefore reflects variation across trained checkpoints and their evaluation runs rather than repeated timing of a single checkpoint.

Strict verification and numerical execution. Strict verification refers to exact token-ID agreement with the target argmax produced by the block-verification forward pass. Because BF16 blockwise and serial execution can diverge numerically, answer accuracy is reported separately; Appendix D provides the full numerical diagnostic.

Implementation environment. Training and evaluation use BF16 precision, inference uses scaled dot-product attention (SDPA), and each worker uses a single GPU. The recorded software environment uses PyTorch 2.4.1 with CUDA 12.1 support and Transformers 4.57.1 (Paszke et al., 2019; Wolf et al., 2020). Runs were distributed across remote GPU workers, and the accelerator model was not recorded consistently on a per-run basis. The timing results therefore characterize this mixed-worker setting rather than one specific accelerator configuration.

B.4 Runtime State and Cache Management

The target model and the recurrent drafter maintain distinct runtime states. The target KV cache is authoritative: it represents only the token prefix that has been committed by target verification. The recurrent transition T_ϕ maintains its own drafting context and is used only to construct speculative candidates.

Recurrent rollout state. At the beginning of a neural drafting round, the transition is conditioned on target-computed hidden states corresponding to the committed prefix. It then recursively predicts

$$\hat{h}_{t+1}, \hat{h}_{t+2}, \ldots, \hat{h}_{t+K},$$

where each newly predicted state is appended to the recurrent drafting context and reused by the next transition step. These predicted states are speculative and are not treated as committed target states.

Target verification and rollback. The target model verifies the anchor and candidate sequence in a causal block forward pass using its existing KV cache. Only the longest consecutively accepted candidate prefix is committed. After verification, the target cache retains the committed prefix, while target computations associated exclusively with rejected speculative positions are discarded.

The recurrent latent states produced during the corresponding neural rollout are also temporary. They are not allowed to persist as authoritative states across speculative rounds. Before the next neural rollout, the recurrent context is synchronized with target-computed hidden states associated with the verified prefix. Consequently, prediction errors may accumulate within one recurrent rollout, but they do not propagate indefinitely across the complete generated sequence.

Interaction with contextual reuse. Contextual lookup and recurrent latent drafting are alternative proposal sources. A lookup hit is verified directly, even when it supplies fewer than the maximum eight tokens; the recurrent transition is not invoked to complete the same proposal. If several consecutive rounds use lookup, neural drafting remains inactive during those rounds. Target-computed boundary states for newly committed tokens are buffered, and the recurrent cache is synchronized lazily before neural drafting is next required. Rejected lookup or neural tokens are never inserted into either the authoritative target history or the lookup index.

Next-round anchor. The verification forward pass also determines the target distribution at the first uncommitted position. The corresponding target-selected token becomes the next anchor, so the decoder does not require an additional single-token corrective target forward solely to obtain that token. This reuse is part of the end-to-end decoding path measured in our experiments.

This cache design separates speculative latent computation from authoritative target state. Predicted hidden states are therefore used to reduce proposal cost without being permanently inserted into the target trajectory. Target verification determines which tokens are committed, and subsequent recurrent rollouts are periodically re-anchored to target-computed representations.

C Additional Experimental Results

This section expands the ablations summarized in the main text with complete three-seed Qwen results. We first compare model scale and training regime, then report proposal-source and recurrent-calibration breakdowns, followed by inference-time sensitivity and response-length results. Appendix A.3 contains the one-step initialization diagnostic, and Appendix D contains the finite-precision verification audit.

C.1 Results across Model Scales and Training Regimes

Tables 9 and 10 give the complete three-seed Hybrid results for Qwen3-4B and Qwen3-8B. These tables include all seven benchmarks as well as the full-joint training branches used for the training-regime comparison.

At Qwen3-4B scale, the Staged models achieve $1.42\times$–$1.49\times$ Hybrid wall-clock speedup, while Joint calibration reaches $1.46\times$–$1.50\times$. At Qwen3-8B scale, Staged reaches $1.54\times$–$1.63\times$, Joint

Table 9: Complete three-seed Hybrid results for Qwen3-4B across training regimes. Values are mean ± sample standard deviation over seeds 42, 43, and 44.

Training	Dataset	Greedy Acc.	Hybrid Acc.	Wall	Throughput
Staged	MATH-500	70.67 ± 0.42	71.07 ± 0.31	$1.42 \pm 0.01\times$	$1.42 \pm 0.01\times$
Staged	AIME 2024	4.44 ± 1.92	3.33 ± 0.00	$1.47 \pm 0.01\times$	$1.47 \pm 0.01\times$
Staged	AIME 2025	4.44 ± 1.92	4.44 ± 1.92	$1.49 \pm 0.03\times$	$1.46 \pm 0.02\times$
Staged	AIME 2026	2.22 ± 1.92	2.22 ± 1.92	$1.45 \pm 0.02\times$	$1.44 \pm 0.03\times$
Staged	AMC 2023	39.17 ± 2.89	39.17 ± 1.44	$1.44 \pm 0.02\times$	$1.44 \pm 0.02\times$
Staged	Minerva/OCW	28.43 ± 0.77	27.82 ± 1.18	$1.44 \pm 0.02\times$	$1.44 \pm 0.01\times$
Staged	OlympiadBench	28.25 ± 0.37	27.60 ± 0.91	$1.46 \pm 0.01\times$	$1.46 \pm 0.01\times$
Joint calibration	MATH-500	70.67 ± 0.42	70.07 ± 1.03	$1.46 \pm 0.01\times$	$1.47 \pm 0.00\times$
Joint calibration	AIME 2024	4.44 ± 1.92	3.33 ± 0.00	$1.50 \pm 0.06\times$	$1.51 \pm 0.04\times$
Joint calibration	AIME 2025	4.44 ± 1.92	4.44 ± 3.85	$1.48 \pm 0.04\times$	$1.48 \pm 0.03\times$
Joint calibration	AIME 2026	2.22 ± 1.92	2.22 ± 1.92	$1.46 \pm 0.01\times$	$1.46 \pm 0.02\times$
Joint calibration	AMC 2023	39.17 ± 2.89	39.17 ± 2.89	$1.48 \pm 0.02\times$	$1.47 \pm 0.01\times$
Joint calibration	Minerva/OCW	28.43 ± 0.77	27.82 ± 2.50	$1.47 \pm 0.02\times$	$1.48 \pm 0.01\times$
Joint calibration	OlympiadBench	28.25 ± 0.37	27.46 ± 0.45	$1.50 \pm 0.01\times$	$1.49 \pm 0.01\times$
Full-joint	MATH-500	70.87 ± 0.42	70.07 ± 1.01	$1.41 \pm 0.01\times$	$1.41 \pm 0.00\times$
Full-joint	AIME 2024	3.33 ± 0.00	5.56 ± 3.85	$1.43 \pm 0.02\times$	$1.45 \pm 0.01\times$
Full-joint	AIME 2025	4.44 ± 3.85	6.67 ± 0.00	$1.43 \pm 0.02\times$	$1.43 \pm 0.02\times$
Full-joint	AIME 2026	2.22 ± 1.92	0.00 ± 0.00	$1.40 \pm 0.01\times$	$1.40 \pm 0.01\times$
Full-joint	AMC 2023	39.17 ± 1.44	39.17 ± 3.82	$1.41 \pm 0.02\times$	$1.41 \pm 0.02\times$
Full-joint	Minerva/OCW	27.70 ± 0.93	26.96 ± 1.66	$1.41 \pm 0.02\times$	$1.41 \pm 0.00\times$
Full-joint	OlympiadBench	28.10 ± 1.20	27.65 ± 0.43	$1.44 \pm 0.01\times$	$1.43 \pm 0.01\times$

Table 10: Complete three-seed Hybrid results for Qwen3-8B across training regimes. Values are mean ± sample standard deviation over seeds 42, 43, and 44. Extended calibration uses additional training budget.

Training	Dataset	Greedy Acc.	Hybrid Acc.	Wall	Throughput
Staged	MATH-500	66.60 ± 0.92	66.27 ± 0.42	$1.56 \pm 0.00\times$	$1.57 \pm 0.01\times$
Staged	AIME 2024	4.44 ± 1.92	3.33 ± 0.00	$1.63 \pm 0.02\times$	$1.64 \pm 0.02\times$
Staged	AIME 2025	4.44 ± 1.92	4.44 ± 1.92	$1.61 \pm 0.01\times$	$1.62 \pm 0.02\times$
Staged	AIME 2026	1.11 ± 1.92	0.00 ± 0.00	$1.58 \pm 0.01\times$	$1.58 \pm 0.02\times$
Staged	AMC 2023	27.50 ± 4.33	26.67 ± 1.44	$1.61 \pm 0.02\times$	$1.59 \pm 0.02\times$
Staged	Minerva/OCW	25.61 ± 0.42	25.61 ± 1.39	$1.54 \pm 0.01\times$	$1.55 \pm 0.01\times$
Staged	OlympiadBench	23.51 ± 0.75	22.81 ± 1.29	$1.61 \pm 0.01\times$	$1.61 \pm 0.01\times$
Joint calibration	MATH-500	66.60 ± 0.92	67.60 ± 0.60	$1.61 \pm 0.01\times$	$1.62 \pm 0.01\times$
Joint calibration	AIME 2024	4.44 ± 1.92	5.56 ± 1.92	$1.70 \pm 0.06\times$	$1.70 \pm 0.02\times$
Joint calibration	AIME 2025	4.44 ± 1.92	3.33 ± 3.33	$1.66 \pm 0.01\times$	$1.65 \pm 0.02\times$
Joint calibration	AIME 2026	1.11 ± 1.92	0.00 ± 0.00	$1.63 \pm 0.02\times$	$1.63 \pm 0.02\times$
Joint calibration	AMC 2023	27.50 ± 4.33	30.00 ± 0.00	$1.67 \pm 0.07\times$	$1.64 \pm 0.05\times$
Joint calibration	Minerva/OCW	25.61 ± 0.42	25.61 ± 1.85	$1.57 \pm 0.01\times$	$1.59 \pm 0.01\times$
Joint calibration	OlympiadBench	23.51 ± 0.75	23.80 ± 0.60	$1.65 \pm 0.02\times$	$1.64 \pm 0.01\times$
Full-joint	MATH-500	68.33 ± 0.46	68.07 ± 0.12	$1.53 \pm 0.01\times$	$1.54 \pm 0.01\times$
Full-joint	AIME 2024	3.33 ± 3.33	4.44 ± 1.92	$1.70 \pm 0.04\times$	$1.68 \pm 0.04\times$
Full-joint	AIME 2025	6.67 ± 0.00	5.56 ± 1.92	$1.63 \pm 0.02\times$	$1.63 \pm 0.02\times$
Full-joint	AIME 2026	0.00 ± 0.00	0.00 ± 0.00	$1.60 \pm 0.00\times$	$1.60 \pm 0.03\times$
Full-joint	AMC 2023	35.83 ± 2.89	36.67 ± 5.20	$1.61 \pm 0.02\times$	$1.60 \pm 0.01\times$
Full-joint	Minerva/OCW	24.63 ± 0.64	25.86 ± 1.29	$1.49 \pm 0.00\times$	$1.49 \pm 0.01\times$
Full-joint	OlympiadBench	25.58 ± 0.48	24.74 ± 0.59	$1.60 \pm 0.01\times$	$1.60 \pm 0.01\times$
Extended calibration	MATH-500	66.60 ± 0.92	67.27 ± 0.12	$1.63 \pm 0.01\times$	$1.63 \pm 0.01\times$
Extended calibration	AIME 2024	4.44 ± 1.92	3.33 ± 0.00	$1.70 \pm 0.05\times$	$1.70 \pm 0.02\times$
Extended calibration	AIME 2025	4.44 ± 1.92	4.44 ± 1.92	$1.65 \pm 0.01\times$	$1.65 \pm 0.01\times$
Extended calibration	AIME 2026	1.11 ± 1.92	1.11 ± 1.92	$1.64 \pm 0.05\times$	$1.64 \pm 0.04\times$
Extended calibration	AMC 2023	27.50 ± 4.33	29.17 ± 1.44	$1.67 \pm 0.04\times$	$1.66 \pm 0.03\times$
Extended calibration	Minerva/OCW	25.61 ± 0.42	26.72 ± 1.39	$1.59 \pm 0.02\times$	$1.60 \pm 0.01\times$
Extended calibration	OlympiadBench	23.51 ± 0.75	23.11 ± 0.90	$1.66 \pm 0.02\times$	$1.66 \pm 0.02\times$

calibration reaches $1.57\times$–$1.70\times$, and Extended calibration reaches $1.59\times$–$1.70\times$. These tables document scale and training-regime effects; the controlled component comparisons are reported separately below.

C.1.1 Equal-budget full-joint comparison.

The staged-to-joint and full-joint Qwen branches both have a 64,023-step endpoint. Table 11 compares their Hybrid wall-clock speedups at this matched cumulative update budget. The staged-to-joint recipe has the higher mean on all seven Qwen3-4B benchmarks; for Qwen3-8B it is higher on six

Table 11: Equal-budget comparison between the default staged-to-joint recipe and full-joint training at 64,023 cumulative optimization steps. The two regimes do not share the same target LoRA trajectory, so this table compares complete training recipes rather than isolating a single causal factor.

Dataset	Q4 Joint calibration	Q4 Full-joint	Q8 Joint calibration	Q8 Full-joint
MATH-500	$1.46 \pm 0.01\times$	$1.41 \pm 0.01\times$	$1.61 \pm 0.01\times$	$1.53 \pm 0.01\times$
AIME 2024	$1.50 \pm 0.06\times$	$1.43 \pm 0.02\times$	$1.70 \pm 0.06\times$	$1.70 \pm 0.04\times$
AIME 2025	$1.48 \pm 0.04\times$	$1.43 \pm 0.02\times$	$1.66 \pm 0.01\times$	$1.63 \pm 0.02\times$
AIME 2026	$1.46 \pm 0.01\times$	$1.40 \pm 0.01\times$	$1.63 \pm 0.02\times$	$1.60 \pm 0.00\times$
AMC 2023	$1.48 \pm 0.02\times$	$1.41 \pm 0.02\times$	$1.67 \pm 0.07\times$	$1.61 \pm 0.02\times$
Minerva/OCW	$1.47 \pm 0.02\times$	$1.41 \pm 0.02\times$	$1.57 \pm 0.01\times$	$1.49 \pm 0.00\times$
OlympiadBench	$1.50 \pm 0.01\times$	$1.44 \pm 0.01\times$	$1.65 \pm 0.02\times$	$1.60 \pm 0.01\times$

Table 12: Proposal-source ablation for Qwen3-4B. Values are three-seed wall-clock speedups relative to paired greedy decoding.

Training	Dataset	Latent-only	Lookup-only	Hybrid
Staged	MATH-500	$1.00 \pm 0.01\times$	$1.46 \pm 0.01\times$	$1.42 \pm 0.01\times$
Staged	AIME 2024	$0.98 \pm 0.03\times$	$1.51 \pm 0.03\times$	$1.47 \pm 0.01\times$
Staged	AIME 2025	$0.99 \pm 0.02\times$	$1.52 \pm 0.02\times$	$1.49 \pm 0.03\times$
Staged	AIME 2026	$0.98 \pm 0.01\times$	$1.48 \pm 0.00\times$	$1.45 \pm 0.02\times$
Staged	AMC 2023	$1.01 \pm 0.02\times$	$1.49 \pm 0.03\times$	$1.44 \pm 0.02\times$
Staged	Minerva/OCW	$1.00 \pm 0.01\times$	$1.48 \pm 0.02\times$	$1.44 \pm 0.02\times$
Staged	OlympiadBench	$1.00 \pm 0.01\times$	$1.51 \pm 0.01\times$	$1.46 \pm 0.01\times$
Joint calibration	MATH-500	$1.07 \pm 0.01\times$	$1.46 \pm 0.01\times$	$1.46 \pm 0.01\times$
Joint calibration	AIME 2024	$1.04 \pm 0.03\times$	$1.51 \pm 0.03\times$	$1.50 \pm 0.06\times$
Joint calibration	AIME 2025	$1.05 \pm 0.02\times$	$1.52 \pm 0.02\times$	$1.48 \pm 0.04\times$
Joint calibration	AIME 2026	$1.04 \pm 0.01\times$	$1.48 \pm 0.01\times$	$1.46 \pm 0.01\times$
Joint calibration	AMC 2023	$1.07 \pm 0.01\times$	$1.49 \pm 0.02\times$	$1.48 \pm 0.02\times$
Joint calibration	Minerva/OCW	$1.04 \pm 0.01\times$	$1.48 \pm 0.02\times$	$1.47 \pm 0.02\times$
Joint calibration	OlympiadBench	$1.05 \pm 0.01\times$	$1.51 \pm 0.01\times$	$1.50 \pm 0.01\times$

benchmarks at the displayed precision and tied on AIME 2024 after rounding. Because the two branches optimize different target LoRA trajectories, this comparison evaluates complete training recipes rather than isolating the causal effect of a single loss term.

C.2 Ablation on Proposal Sources

The main text reports the Qwen3-8B Extended calibration proposal-source ablation. Tables 12 and 13 provide the full three-seed breakdown across training stages. Latent-only isolates the learned target-derived transition, Lookup-only isolates contextual reuse, and Hybrid uses lookup when a reusable continuation is available and otherwise falls back to the recurrent transition.

The scale comparison reinforces the separation between proposal quality and proposal cost. At Qwen3-4B scale, the staged Latent-only route is approximately break-even, but recurrent calibration moves it to a consistent net speedup. At Qwen3-8B scale, Latent-only is already beneficial before joint calibration and improves further with continued recurrent optimization. Under Extended calibration, Hybrid also exceeds Lookup-only across the full benchmark suite, showing that neural drafting contributes useful proposals outside the contexts covered by reuse.

C.3 Detailed Recurrent-Calibration Results

The main-text table reports the mean trajectories compactly. Tables 14 and 15 provide the corresponding three-seed means and sample standard deviations for every benchmark. Within each staged-to-calibrated comparison, the target checkpoint is unchanged and only the transition and boundary readout are updated.

Across both model scales, recurrent calibration most consistently improves the learned neural drafter itself: Latent-only acceptance and Latent-only wall-clock speedup increase across the full benchmark suite. The Hybrid route follows the same overall direction but is less strictly monotonic because its total latency also depends on how often contextual reuse is available and how many lookup tokens are accepted.

Table 13: Proposal-source ablation for Qwen3-8B. Values are three-seed wall-clock speedups relative to paired greedy decoding. Extended calibration uses additional training budget.

Training	Dataset	Latent-only	Lookup-only	Hybrid
Staged	MATH-500	$1.16 \pm 0.00\times$	$1.51 \pm 0.01\times$	$1.56 \pm 0.00\times$
Staged	AIME 2024	$1.12 \pm 0.03\times$	$1.63 \pm 0.01\times$	$1.63 \pm 0.02\times$
Staged	AIME 2025	$1.11 \pm 0.01\times$	$1.59 \pm 0.02\times$	$1.61 \pm 0.01\times$
Staged	AIME 2026	$1.13 \pm 0.00\times$	$1.57 \pm 0.04\times$	$1.58 \pm 0.01\times$
Staged	AMC 2023	$1.18 \pm 0.00\times$	$1.56 \pm 0.03\times$	$1.61 \pm 0.02\times$
Staged	Minerva/OCW	$1.13 \pm 0.01\times$	$1.50 \pm 0.01\times$	$1.54 \pm 0.01\times$
Staged	OlympiadBench	$1.14 \pm 0.01\times$	$1.56 \pm 0.01\times$	$1.61 \pm 0.01\times$
Joint calibration	MATH-500	$1.24 \pm 0.01\times$	$1.51 \pm 0.01\times$	$1.61 \pm 0.01\times$
Joint calibration	AIME 2024	$1.18 \pm 0.03\times$	$1.64 \pm 0.01\times$	$1.70 \pm 0.06\times$
Joint calibration	AIME 2025	$1.18 \pm 0.01\times$	$1.58 \pm 0.02\times$	$1.66 \pm 0.01\times$
Joint calibration	AIME 2026	$1.20 \pm 0.01\times$	$1.57 \pm 0.04\times$	$1.63 \pm 0.02\times$
Joint calibration	AMC 2023	$1.25 \pm 0.01\times$	$1.56 \pm 0.03\times$	$1.67 \pm 0.07\times$
Joint calibration	Minerva/OCW	$1.18 \pm 0.01\times$	$1.50 \pm 0.01\times$	$1.57 \pm 0.01\times$
Joint calibration	OlympiadBench	$1.21 \pm 0.01\times$	$1.56 \pm 0.01\times$	$1.65 \pm 0.02\times$
Extended calibration	MATH-500	$1.27 \pm 0.01\times$	$1.51 \pm 0.01\times$	$1.63 \pm 0.01\times$
Extended calibration	AIME 2024	$1.21 \pm 0.04\times$	$1.64 \pm 0.01\times$	$1.70 \pm 0.05\times$
Extended calibration	AIME 2025	$1.20 \pm 0.01\times$	$1.59 \pm 0.02\times$	$1.65 \pm 0.01\times$
Extended calibration	AIME 2026	$1.22 \pm 0.01\times$	$1.56 \pm 0.05\times$	$1.64 \pm 0.05\times$
Extended calibration	AMC 2023	$1.27 \pm 0.01\times$	$1.56 \pm 0.04\times$	$1.67 \pm 0.04\times$
Extended calibration	Minerva/OCW	$1.20 \pm 0.01\times$	$1.50 \pm 0.01\times$	$1.59 \pm 0.02\times$
Extended calibration	OlympiadBench	$1.23 \pm 0.01\times$	$1.56 \pm 0.01\times$	$1.66 \pm 0.02\times$

Table 14: Full three-seed Qwen3-4B recurrent-calibration breakdown. Acceptance is measured on the Latent-only route.

Dataset	Training stage	Latent accept. (%)	Latent-only wall	Hybrid wall
MATH-500	Staged	12.22 ± 0.38	$1.00 \pm 0.01\times$	$1.42 \pm 0.01\times$
	Joint calibration	16.42 ± 0.56	$1.07 \pm 0.01\times$	$1.46 \pm 0.01\times$
AIME 2024	Staged	10.65 ± 0.77	$0.98 \pm 0.03\times$	$1.47 \pm 0.01\times$
	Joint calibration	14.00 ± 0.69	$1.04 \pm 0.03\times$	$1.50 \pm 0.06\times$
AIME 2025	Staged	9.98 ± 0.64	$0.99 \pm 0.02\times$	$1.49 \pm 0.03\times$
	Joint calibration	13.66 ± 0.82	$1.05 \pm 0.02\times$	$1.48 \pm 0.04\times$
AIME 2026	Staged	10.74 ± 0.67	$0.98 \pm 0.01\times$	$1.45 \pm 0.02\times$
	Joint calibration	14.19 ± 0.70	$1.04 \pm 0.01\times$	$1.46 \pm 0.01\times$
AMC 2023	Staged	12.38 ± 0.38	$1.01 \pm 0.02\times$	$1.44 \pm 0.02\times$
	Joint calibration	16.31 ± 0.61	$1.07 \pm 0.01\times$	$1.48 \pm 0.02\times$
Minerva/OCW	Staged	11.58 ± 0.41	$1.00 \pm 0.01\times$	$1.44 \pm 0.02\times$
	Joint calibration	14.86 ± 0.72	$1.04 \pm 0.01\times$	$1.47 \pm 0.02\times$
OlympiadBench	Staged	11.36 ± 0.46	$1.00 \pm 0.01\times$	$1.46 \pm 0.01\times$
	Joint calibration	15.13 ± 0.56	$1.05 \pm 0.01\times$	$1.50 \pm 0.01\times$

C.4 Inference-Time Sensitivity

The main text summarizes the effects of verification threshold and speculation start position. Table 16 reports the complete three-seed Qwen3-8B Extended calibration grid. Relaxing λ from 1.00 to 0.90 changes speed modestly and in a task-dependent way. Delaying speculation from token 32 to token 128 reduces wall-clock speedup on every benchmark under both thresholds because a smaller fraction of the response is eligible for speculative decoding.

C.5 Effect of Generation Length

We additionally evaluate a maximum response length of 4096 tokens. Complete three-seed coverage is currently available for 16 Qwen3-8B training–dataset configurations: AIME 2024, AIME 2025, AIME 2026, and AMC 2023 under each of Staged, Joint calibration, Full-joint, and Extended calibration. MATH-500, Minerva/OCW, and OlympiadBench are omitted because complete three-seed coverage is not yet available.

Across these 16 completed configurations, Latent-only wall-clock speedup ranges from $1.06\times$ to $1.40\times$, Lookup-only from $2.09\times$ to $2.58\times$, and Hybrid from $2.05\times$ to $2.57\times$. These larger measured speedups are consistent with longer generations providing more opportunities to amortize drafting

Table 15: Full three-seed Qwen3-8B recurrent-calibration breakdown. Acceptance is measured on the Latent-only route. Extended calibration uses additional training budget.

Dataset	Training stage	Latent accept. (%)	Latent-only wall	Hybrid wall
MATH-500	Staged	19.44 ± 0.20	$1.16 \pm 0.00\times$	$1.56 \pm 0.00\times$
	Joint calibration	24.66 ± 0.33	$1.24 \pm 0.01\times$	$1.61 \pm 0.01\times$
	Extended calibration	26.68 ± 0.13	$1.27 \pm 0.01\times$	$1.63 \pm 0.01\times$
AIME 2024	Staged	16.45 ± 0.46	$1.12 \pm 0.03\times$	$1.63 \pm 0.02\times$
	Joint calibration	20.54 ± 0.24	$1.18 \pm 0.03\times$	$1.70 \pm 0.06\times$
	Extended calibration	22.17 ± 0.62	$1.21 \pm 0.04\times$	$1.70 \pm 0.05\times$
AIME 2025	Staged	16.43 ± 0.62	$1.11 \pm 0.01\times$	$1.61 \pm 0.01\times$
	Joint calibration	20.59 ± 0.67	$1.18 \pm 0.01\times$	$1.66 \pm 0.01\times$
	Extended calibration	21.95 ± 0.50	$1.20 \pm 0.01\times$	$1.65 \pm 0.01\times$
AIME 2026	Staged	17.12 ± 0.23	$1.13 \pm 0.00\times$	$1.58 \pm 0.01\times$
	Joint calibration	21.02 ± 0.35	$1.20 \pm 0.01\times$	$1.63 \pm 0.02\times$
	Extended calibration	22.40 ± 0.30	$1.22 \pm 0.01\times$	$1.64 \pm 0.05\times$
AMC 2023	Staged	19.40 ± 0.13	$1.18 \pm 0.00\times$	$1.61 \pm 0.02\times$
	Joint calibration	23.79 ± 0.60	$1.25 \pm 0.01\times$	$1.67 \pm 0.07\times$
	Extended calibration	25.29 ± 0.21	$1.27 \pm 0.01\times$	$1.67 \pm 0.04\times$
Minerva/OCW	Staged	18.08 ± 0.11	$1.13 \pm 0.01\times$	$1.54 \pm 0.01\times$
	Joint calibration	21.20 ± 0.42	$1.18 \pm 0.01\times$	$1.57 \pm 0.01\times$
	Extended calibration	22.62 ± 0.27	$1.20 \pm 0.01\times$	$1.59 \pm 0.02\times$
OlympiadBench	Staged	17.83 ± 0.27	$1.14 \pm 0.01\times$	$1.61 \pm 0.01\times$
	Joint calibration	22.07 ± 0.26	$1.21 \pm 0.01\times$	$1.65 \pm 0.02\times$
	Extended calibration	23.49 ± 0.30	$1.23 \pm 0.01\times$	$1.66 \pm 0.02\times$

Table 16: Three-seed inference-time sensitivity for Qwen3-8B Extended calibration. Each configuration reports Hybrid final-answer accuracy and wall-clock speedup; greedy accuracy is shown once because the paired greedy checkpoint is unchanged across the four settings.

Dataset	Metric	Start32/1.00	Start32/0.90	Start128/1.00	Start128/0.90
MATH-500	Hybrid Acc.	67.27 ± 0.12	67.27 ± 0.64	66.60 ± 0.87	66.47 ± 1.30
	Wall speedup	$1.63 \pm 0.01\times$	$1.64 \pm 0.01\times$	$1.51 \pm 0.01\times$	$1.52 \pm 0.01\times$
AIME 2024	Hybrid Acc.	3.33 ± 0.00	5.56 ± 1.92	4.44 ± 1.92	4.44 ± 1.92
	Wall speedup	$1.70 \pm 0.05\times$	$1.70 \pm 0.05\times$	$1.64 \pm 0.08\times$	$1.65 \pm 0.03\times$
AIME 2025	Hybrid Acc.	4.44 ± 1.92	4.44 ± 3.85	3.33 ± 3.33	2.22 ± 1.92
	Wall speedup	$1.65 \pm 0.01\times$	$1.68 \pm 0.02\times$	$1.60 \pm 0.04\times$	$1.60 \pm 0.03\times$
AIME 2026	Hybrid Acc.	1.11 ± 1.92	2.22 ± 1.92	2.22 ± 1.92	2.22 ± 1.92
	Wall speedup	$1.64 \pm 0.05\times$	$1.66 \pm 0.03\times$	$1.55 \pm 0.01\times$	$1.58 \pm 0.04\times$
AMC 2023	Hybrid Acc.	29.17 ± 1.44	31.67 ± 5.20	27.50 ± 5.00	30.00 ± 2.50
	Wall speedup	$1.67 \pm 0.04\times$	$1.69 \pm 0.03\times$	$1.57 \pm 0.04\times$	$1.59 \pm 0.04\times$
Minerva/OCW	Hybrid Acc.	26.72 ± 1.39	27.08 ± 0.93	25.98 ± 0.21	27.08 ± 0.93
	Wall speedup	$1.59 \pm 0.02\times$	$1.58 \pm 0.01\times$	$1.50 \pm 0.00\times$	$1.50 \pm 0.01\times$
OlympiadBench	Hybrid Acc.	23.11 ± 0.90	22.57 ± 1.63	23.60 ± 0.52	23.60 ± 0.31
	Wall speedup	$1.66 \pm 0.02\times$	$1.68 \pm 0.02\times$	$1.58 \pm 0.01\times$	$1.59 \pm 0.01\times$

Greedy accuracies (mean $\pm$ sample standard deviation) are: MATH-500 66.60 ± 0.92, AIME 2024 4.44 ± 1.92, AIME 2025 4.44 ± 1.92, AIME 2026 1.11 ± 1.92, AMC 2023 27.50 ± 4.33, Minerva/OCW 25.61 ± 0.42, and OlympiadBench 23.51 ± 0.75.

and verification overhead. Because benchmark coverage is incomplete, we interpret this comparison as a response-length sensitivity analysis rather than a complete long-generation evaluation.

D Numerical Behavior under Strict Verification

Our default setting uses strict verification with $\lambda = 1.00$. In this mode, a candidate is accepted only when its token ID equals the target-model argmax produced by the corresponding verification forward pass. The greedy path and the strict verification path use the same PyTorch `argmax` rule, including the same tie-breaking behavior.

Strict verification is defined by token-ID agreement with the target argmax produced by the verification forward pass; the definition does not extend to bitwise identity between blockwise and serial execution. In BF16, the two execution shapes can invoke different GPU matrix kernels and therefore produce slightly different floating-point results.

Table 17: Detailed Qwen3-8B results under the 4096-token response budget for all training–dataset configurations with complete three-seed coverage. Values are mean ± sample standard deviation. MATH-500, Minerva/OCW, and OlympiadBench are omitted because complete three-seed coverage is not yet available.

Training	Dataset	Greedy Acc.	Hybrid Acc.	Latent-only	Lookup-only	Hybrid
Staged	AIME 2024	15.56 ± 1.92	18.89 ± 3.85	$1.08 \pm 0.15\times$	$2.50 \pm 0.16\times$	$2.48 \pm 0.11\times$
Staged	AIME 2025	15.56 ± 1.92	11.11 ± 3.85	$1.06 \pm 0.06\times$	$2.09 \pm 0.14\times$	$2.18 \pm 0.11\times$
Staged	AIME 2026	11.11 ± 5.09	6.67 ± 3.33	$1.17 \pm 0.06\times$	$2.16 \pm 0.18\times$	$2.05 \pm 0.07\times$
Staged	AMC 2023	41.67 ± 3.82	40.83 ± 2.89	$1.26 \pm 0.09\times$	$2.21 \pm 0.31\times$	$2.24 \pm 0.20\times$
Joint calibration	AIME 2024	15.56 ± 1.92	16.67 ± 0.00	$1.18 \pm 0.21\times$	$2.50 \pm 0.17\times$	$2.55 \pm 0.26\times$
Joint calibration	AIME 2025	15.56 ± 1.92	15.56 ± 3.85	$1.11 \pm 0.06\times$	$2.09 \pm 0.14\times$	$2.25 \pm 0.09\times$
Joint calibration	AIME 2026	11.11 ± 5.09	11.11 ± 3.85	$1.22 \pm 0.09\times$	$2.17 \pm 0.19\times$	$2.26 \pm 0.09\times$
Joint calibration	AMC 2023	41.67 ± 3.82	48.33 ± 3.82	$1.35 \pm 0.06\times$	$2.21 \pm 0.32\times$	$2.40 \pm 0.32\times$
Full-joint	AIME 2024	17.78 ± 5.09	17.78 ± 1.92	$1.13 \pm 0.01\times$	$2.58 \pm 0.04\times$	$2.57 \pm 0.07\times$
Full-joint	AIME 2025	14.44 ± 1.92	13.33 ± 0.00	$1.13 \pm 0.14\times$	$2.44 \pm 0.03\times$	$2.49 \pm 0.16\times$
Full-joint	AIME 2026	11.11 ± 3.85	8.89 ± 1.92	$1.17 \pm 0.11\times$	$2.33 \pm 0.05\times$	$2.33 \pm 0.03\times$
Full-joint	AMC 2023	55.00 ± 2.50	53.33 ± 6.29	$1.20 \pm 0.07\times$	$2.16 \pm 0.03\times$	$2.08 \pm 0.04\times$
Extended calibration	AIME 2024	15.56 ± 1.92	11.11 ± 5.09	$1.18 \pm 0.12\times$	$2.49 \pm 0.17\times$	$2.53 \pm 0.31\times$
Extended calibration	AIME 2025	15.56 ± 1.92	11.11 ± 5.09	$1.12 \pm 0.05\times$	$2.09 \pm 0.15\times$	$2.23 \pm 0.02\times$
Extended calibration	AIME 2026	11.11 ± 5.09	7.78 ± 3.85	$1.21 \pm 0.06\times$	$2.17 \pm 0.18\times$	$2.17 \pm 0.14\times$
Extended calibration	AMC 2023	41.67 ± 3.82	44.17 ± 3.82	$1.40 \pm 0.11\times$	$2.21 \pm 0.31\times$	$2.25 \pm 0.20\times$

Table 18: Exact output-sequence agreement between strict speculative decoding and its paired serial greedy run in the diagnostic snapshot.

Model	Route	Pairs	Exact	Agreement
Qwen3-4B	Latent-only	19,035	4,788	25.15%
Qwen3-4B	Lookup-only	19,035	5,037	26.46%
Qwen3-4B	Hybrid	19,035	4,763	25.02%
Qwen3-8B	Latent-only	48,411	11,206	23.15%
Qwen3-8B	Lookup-only	48,411	11,568	23.90%
Qwen3-8B	Hybrid	48,411	11,355	23.46%

We audit saved strict-verification outputs from Qwen3-4B and Qwen3-8B across multiple checkpoints, benchmarks, proposal sources, and speculation starts. The diagnostic snapshot contains 202,338 paired greedy–speculative runs, including repeated evaluations of the same benchmark examples under different checkpoints and decoding configurations.

Across the audited comparisons, every accepted draft token has a corresponding recorded strict block decision, and no accepted token disagrees with that decision. Most first sequence divergences instead arise from a target-selected anchor after a block forward.

To investigate the numerical source, we replayed and instrumented a concrete Qwen3-4B example at its first divergent output position. Immediately before the divergent forward pass, the target KV caches of the greedy and speculative executions are element-wise identical in all 36 Transformer layers. The corresponding query position also has the same causally visible prefix. The principal execution difference is that greedy evaluates one token at a time, whereas the strict verification path evaluates a three-token block.

For the diagnosed position, the two leading greedy logits are 34.50 and 34.25, whereas the corresponding block-forward logits become 34.25 and 34.25. Thus, the greedy computation has a margin of 0.25, while the block computation produces a tie. Both executions use the same argmax rule, so the block path selects the smaller token ID in the tie. The maximum absolute difference over the complete vocabulary logits is 0.25.

Operator-level tracing identifies the earliest observed difference in the first-layer query projection. Its input is element-wise identical between the two executions, but the query shapes trigger different GPU computation paths. In the diagnosed run, the single-token case uses a matrix–vector execution path, whereas the three-token case uses a BF16 matrix–matrix kernel followed by a split reduction. The first query-projection output differs by at most 3.91×10^{-3}, after which the numerical difference propagates through subsequent layers.

We further perform controlled interventions. When all linear layers are evaluated token by token while retaining blockwise attention, the complete vocabulary logits at the diagnosed position become element-wise identical to the original greedy logits. In contrast, evaluating only the final language-model head token by token or evaluating only attention query by query does not remove the divergence. A local FP32 control reduces the maximum difference between the single-token and block logits to approximately 1.05×10^{-5} and both paths select the same top-1 token, although the full logit vectors are still not bitwise identical.

These interventions establish the cause of the diagnosed divergence: the difference is produced by shape-dependent finite-precision matrix computation rather than by a different verification rule or by accepting a speculative token without a corresponding strict block decision. We do not assume that this operator-level explanation has been individually established for every divergent sequence in Table 18; the detailed intervention is currently available for the replayed diagnostic state.

Accordingly, throughout this paper, *strict verification* means exact token-ID agreement with the target argmax produced by the verification forward pass; it does not extend to bitwise identity between blockwise and separately executed serial greedy computation. This distinction also explains why paired greedy and $\lambda = 1.00$ speculative runs can occasionally have different final-answer accuracy despite using the same target checkpoint and strict candidate acceptance.

E Qualitative Analysis and Case Studies

To complement the aggregate acceptance and speed measurements, we inspect representative decoding trajectories that illustrate three mechanisms: consecutive acceptance of recurrent latent proposals, switching between contextual reuse and neural drafting, and recovery after partial rejection. All cases use Qwen3-8B after Extended calibration with training seed 42 on MATH-500, under strict verification with $\lambda = 1.00$, speculation start position 32, a maximum prompt length of 4096 tokens, and a maximum response length of 1024 tokens.

Case selection. Examples are selected by deterministic behavioral criteria rather than final-answer correctness or single-example latency. Case 1 selects the example closest to the median per-example Latent-only acceptance rate among trajectories containing two consecutive $2/2$ neural-acceptance rounds, yielding sample 284 with a 25.04% acceptance rate. Case 2 selects the first Hybrid trajectory containing a lookup–neural–lookup pattern in which both proposal routes contribute accepted drafts, yielding sample 0. Case 3 selects the first Latent-only trajectory with adjacent $1/2$ then $2/2$ neural-acceptance rounds, again yielding sample 0 under a different decoding route.

Trace interpretation. The logged `proposed_len` and `accepted_len` values include the target-generated anchor; proposal and acceptance counts below therefore refer only to draft tokens. The logs retain committed tokens but not the identities of rejected neural proposals, which are marked as unavailable. For the Hybrid example, lookup hit/miss status is obtained by deterministic replay of the frozen contextual-lookup procedure using the original prompt, tokenizer, and committed prefix.

Case 1: Consecutive Recurrent Latent Acceptance

Setup. MATH-500 sample 284, Latent-only decoding. Across the complete example, the recurrent drafter accepts 302 of 1206 neural draft tokens, corresponding to a 25.04% acceptance rate.

Trace.

Round	Anchor ID	Draft IDs	Accepted	Next anchor
33	90	16, 15170	2/2	87
34	87	92, 1124	2/2	1291
35	1291	not saved	0/2	8
36	8	1124, not saved	1/2	1291
37	1291	not saved	0/2	8
38	8	284, 1124	2/2	37018
39	37018	90, 16	2/2	15170
40	15170	not saved	0/2	18

Observation. Rounds 33 and 34 each accept both recurrent draft tokens, so a single target verification advances the committed sequence by three tokens: one target-generated anchor and two neural drafts. The surrounding rounds also contain partial and full rejection, after which full acceptance reappears. The example therefore illustrates how useful multi-token progress can arise intermittently without requiring every recurrent rollout to be correct.

Case 2: Switching between Contextual Reuse and Latent Drafting

Setup. MATH-500 sample 0, Hybrid decoding. The complete trajectory contains 35 lookup blocks and 120 neural blocks, with 131 accepted draft tokens out of 512 proposals.

Trace.

Round	Source	Proposed	Accepted	Lookup state
77	Lookup	8	5	hit
78	Neural	2	1	miss
79	Neural	2	2	miss
80	Neural	2	1	miss
81	Neural	2	1	miss
82	Neural	2	2	miss
83	Lookup	8	1	hit

Observation. Contextual reuse provides a long candidate block at Round 77. During Rounds 78–82, no reusable continuation is available and the decoder falls back to recurrent latent drafting, which continues to produce accepted proposals. Lookup becomes available again at Round 83. The example illustrates that the two proposal routes alternate according to contextual availability rather than duplicating the same proposal process.

Case 3: Partial Rejection and Re-Anchoring

Setup. MATH-500 sample 0, Latent-only decoding. The complete trajectory accepts 101 of 370 proposed neural draft tokens.

Trace.

Round	Anchor ID	Draft IDs	Accepted	Next anchor
37	220	15, not saved	1/2	3
38	3	323, 400	2/2	15

Observation. At Round 37, the first neural draft is accepted and the second is rejected. The rejected suffix is discarded, and the target verification pass selects token 3 as the next anchor. Before the next neural rollout, the drafting context is synchronized to target-computed representations for the committed prefix rather than continuing from the rejected speculative trajectory. At Round 38, both newly proposed draft tokens are accepted.

The three cases are intended to visualize mechanisms already measured quantitatively rather than to provide independent estimates of their frequency or effect size.

F Notation Summary

Tables 19 and 20 summarize the notation used in the main method and evaluation protocol. Symbols introduced only for the local analyses in Appendix A are defined where they appear.

Table 19: Core model and inference notation.

Symbol	Meaning
$x_{\leq t}$	Committed token prefix at decoding position t.
F_θ	Complete target language model used for token prediction and candidate verification.
$p_\theta(v \mid x_{\leq t})$	Target-model next-token distribution.
$H_\theta^{(b)}$	Target computation up to the selected intermediate boundary b.
$h_{\leq t}$	Target-computed hidden-state sequence at boundary b for the committed prefix.
T_ϕ	Recurrent transition initialized from a contiguous interval of target-model Transformer layers.
$\tau_\phi(S)$	One-step transition output $\mathrm{last}[T_\phi(S)]$ for latent sequence S.
$\hat{h}_{t+k}$	Predicted hidden state at rollout step k in the selected representation space.
$\mathcal{I}$	Contiguous target-layer interval copied to initialize T_ϕ.
B_ψ	Boundary readout that maps an intermediate hidden state to vocabulary logits.
$q_\psi(v \mid h)$	Token distribution decoded from hidden state h.
a_t	Target-generated anchor token for the current decoding round.
$d_{t,k}$	Neural draft token read out from $\hat{h}_{t+k}$.
$C_t = (c_{t,1}, \ldots, c_{t,K_t})$	Generic candidate sequence produced by neural drafting or contextual reuse.
K	Maximum recurrent neural drafting horizon.
K_t	Number of candidates proposed in decoding round t.
m_t	Number of consecutively accepted candidates in round t.
$\pi_{t,j}(v)$	Target distribution used to verify candidate $c_{t,j}$.
$r_{t,j}$	Relative target support of candidate $c_{t,j}$.
λ	Verification threshold; $\lambda = 1.00$ gives strict target-argmax matching.

Table 20: Training-objective notation.

Symbol	Meaning
$\mathrm{sg}(\cdot)$	Stop-gradient operator.
$\mathcal{L}_{\mathrm{step}}$	One-step hidden-state prediction objective.
$\ell_\beta(\cdot,\cdot)$	Smooth-L_1 state distance used for recurrent state supervision.
$\mathcal{L}_{\mathrm{state}}$	Multi-step hidden-state regression objective.
$\mathcal{L}_{\mathrm{dist}}$	Distribution-alignment objective between predicted-state readouts and target distributions.
$\mathcal{L}_{\mathrm{joint}}$	Joint recurrent calibration objective.
α	Weight on the one-step objective in joint calibration.
γ	Weight on distribution alignment in the recurrent objective.
$w(s)$	Schedule controlling the recurrent-loss contribution at calibration step s.
β	Smooth-L_1 transition parameter used in ℓ_β; $\beta = 0.1$ in the reported joint runs.
s	Zero-based number of completed optimizer updates within joint calibration.
$\bar{p}_t$	Fixed next-token distribution from the adapted target used during readout initialization.
$\eta_{\mathrm{CE}}, \eta_{\mathrm{KL}}$	Weights on the cross-entropy and distillation terms in readout initialization.