TACG: TRAJECTORY-AWARE COMMIT GATING FOR DIFFUSION LANGUAGE MODEL DECODING

Chengcheng Wang[1*] **Tingzhang Luo**[2*] **Wenhao Li**[1] **Jianyuan Guo**[2] **Chang Xu**[1]
[1]University of Sydney [2]City University of Hong Kong
cwan0785@uni.sydney.edu.au tingzhluo3-c@my.cityu.edu.hk
weli0055@uni.sydney.edu.au jianyguo@cityu.edu.hk c.xu@sydney.edu.au

ABSTRACT

Diffusion language models predict many masked positions at each denoising step, but their efficiency and output quality depend on which predictions enter the evolving context. Confidence is informative but incomplete for this decision: at matched confidence, proposals with different identity and support histories exhibit different one-step revision rates. We propose Trajectory-Aware Commit Gating (TACG), a training-free decoder that separates the current token proposal from the decision to commit it. TACG scores temporal support for the current proposal by contrasting its logits with an exponential moving average of earlier logits, requires the proposal identity to persist through a History Gate, and promotes a capped number of additional positions per step. The gate decides which positions to commit and when; every written token remains the contemporaneous base proposal. On complete four-task LLaDA evaluations under the stated single-seed protocol, measured task-specific operating points reach 51.02% mean accuracy with 67.82 mean model evaluations, compared with 48.00% and 76.59 for Native Confidence. Under a matched per-event commit-count replay, history-guided selection exceeds confidence ranking by 8.3 accuracy points at identical per-sample NFE. History-reference interventions alter commitment timing while preserving the base-proposal write rule. Experiments on Dream and LLaDA2-Mini characterize quality–cost behavior across backbones and generation lengths. The code is publicly available at `https://github.com/Clarence-CV/TACG-DLLM`.

1 INTRODUCTION

Diffusion language models (DLLMs) generate text by repeatedly predicting the values of masked positions (Austin et al., 2023; Sahoo et al., 2024; Nie et al., 2025). A single forward pass proposes tokens at many positions, creating an opportunity for parallel generation. Realizing this opportunity requires a second decision: which proposals should enter the context now? In the non-revising decoders considered here, a committed token stays in the sequence and conditions subsequent predictions. Decoding therefore controls both the proposed token and when that proposal enters the sequence.

Confidence provides a natural rule for this decision, but a scalar threshold uses only the current probability. Raising it delays commitment; lowering it admits more candidates without distinguishing how their support developed. Each masked position also carries a history of proposed identities and logits across denoising steps. Two proposals with the same current confidence may therefore have different trajectories, such as repeated support for the same token or a recent switch to a new candidate. Recent stability-aware decoders such as KLASS exploit changes in the predictive distribution across steps (Kim et al., 2026), while the role of candidate-specific proposal history in the commitment decision remains open.

We first test whether proposal persistence and support trend contain information that current confidence omits. A one-step hold (HOLD) probe keeps a candidate masked for one additional context update and records whether its top-1 identity changes. Figure 1 reports two revision-rate differences

[*]Equal contribution.

on GSM8K. Within confidence bins, recently switched proposals revise more often than persistent proposals. After matching confidence and persistence, weakening proposals revise more often than strengthening proposals. The probe measures one-step proposal revision; it does not measure correctness or calibration.

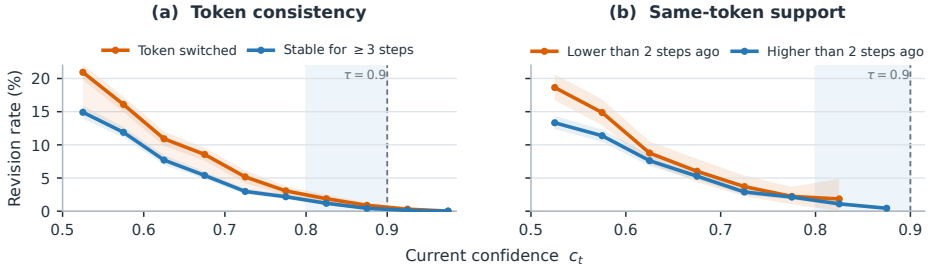


Figure 1: **Revision rates differ at matched confidence**. We delay commitment by one context update on GSM8K and record whether the model changes its top-1 proposal. (a) Within confidence bins, the model revises recently switched proposals more often than proposals that retain the same token for at least three steps. (b) After matching confidence and persistence, the model revises proposals with declining same-token support more often than proposals with increasing support.

This observation motivates a separation between proposing a token and committing its position. We introduce *Trajectory-Aware Commit Gating* (TACG), a training-free decoder that uses two history signals. Figure 2 summarizes the resulting information flow. Temporal Implicit Logits Guidance (TILG) contrasts the current logits with an exponential-moving-average reference and scores the current proposal's support. A History Gate (HG) checks proposal persistence. TACG uses these signals to gate base commitments and rank a capped set of additional positions; every write remains the contemporaneous base proposal.

The separation defines a local write invariant: TACG changes which positions enter the context and when they enter it, while the token written at each accepted position comes from the base decoder at that step. Later context can therefore produce different subsequent proposals and different final text. We evaluate this mechanism with a selection-versus-writing intervention, a matched per-event commit-count replay, and logit-reference interventions.

Our headline LLaDA comparison uses complete HumanEval, MBPP, GSM8K, and MATH500 splits. The reported task-specific TACG operating points are measured on completed full-task runs from the candidate roster under the stated single-seed protocol; they add 3.02 accuracy points and reduce mean model calls by 11.45% relative to Native Confidence. Dream and LLaDA2-Mini extend the analysis to a second backbone and longer generation lengths; we report matched timing and throughput studies with their own settings.

Our contributions are:

1. We formulate parallel DLLM decoding as a commitment problem and introduce TACG, which combines candidate-conditioned temporal support with proposal persistence while retaining the base proposal at every write.

2. We compare selection and writing with a 2×2 intervention and test position choice with a matched per-event commit-count replay. The replay yields 49.42% versus 41.16% accuracy for history-guided versus confidence-ranked selection at identical per-sample NFE.

3. We report complete quality–NFE results on three decoder families, together with logit-reference, sensitivity, timing, and length–batch studies that delimit how the tested gate changes decoding trajectories and cost.

2 BACKGROUND: MASKED DIFFUSION EXPOSES A COMMITMENT DECISION

2.1 THE DECODER CHOOSES WHICH PROPOSALS ENTER THE CONTEXT

Given a masked sequence x_t, a DLLM predicts $p_t(i, v) = p_\theta(x_0^i = v \mid x_t)$ over vocabulary entries v at each masked position i. A model call produces predictions for all active positions in parallel, after which the decoder chooses a subset of proposals to reveal. In the non-revising setting considered here, a revealed token remains in the sequence and becomes context for subsequent model calls. The model therefore supplies the current proposal, while the decoder chooses when to commit that proposal. LLaDA (Nie et al., 2025) and Dream (Ye et al., 2025) instantiate this masked-diffusion setting, while LLaDA2-Mini uses a block-diffusion variant (Bie et al., 2025).

2.2 A PROPOSAL TRAJECTORY CONTAINS DISCRETE AND CONTINUOUS HISTORY

Before the decoder commits a position, repeated denoising calls expose a trajectory of its proposed identity and model support. For position i at step t, let $\hat{x}_{t,i}$ denote the current top-1 proposal and $c_{t,i}$ its confidence. Proposal persistence counts consecutive observations that retain the same top-1 identity. For the current proposal $v = \hat{x}_{t,i}$ and $t \geq 2$, a support-trend diagnostic is

$$g_{t,i} = \log p_t(i, v) - \log p_{t-2}(i, v), \tag{1}$$

where positive values indicate increasing support for the current proposal and negative values indicate decreasing support. This two-step diagnostic is separate from the implemented EMA score b_i in Section 4.

We use a one-step HOLD diagnostic to relate these histories to proposal revision. If the decoder would commit position i at step t, we keep it masked for one additional model call while the decoder updates the surrounding context. We record whether the top-1 proposal changes,

$$R_{t,i} = \mathbb{1}\left[\hat{x}_{t+1,i}^{\text{hold}} \neq \hat{x}_{t,i}\right]. \tag{2}$$

A revision $R_{t,i} = 1$ means that the current proposal does not survive one additional context update. It does not imply that either proposal is correct. We use the HOLD probe only for analysis; it adds no model call during TACG inference.

The comparisons in Figure 1 stratify proposals by current confidence before comparing recently switched and persistent proposals. They then stratify by confidence and persistence before comparing decreasing and increasing support. These comparisons motivate a candidate-conditioned temporal signal, while the method and experiments keep its interpretation operational: it ranks positions for commitment and does not provide a correctness guarantee.

3 RELATED WORK

Parallel commit policies. Fast-dLLM combines confidence-aware selection with approximate caching (Wu et al., 2025). APD adapts parallel sampling to the denoising state (Israel et al., 2025), LocalLeap groups positions using local determinism (Kong et al., 2025), Prophet uses a top-2 confidence gap to trigger early commits (Li et al., 2026), and Dynamic Sliding Block changes the reveal schedule with a dynamically sized sliding block (Luo et al., 2026). These methods change the commit schedule through confidence, local consistency, or a predefined schedule. TACG uses the same commit interface and adds a candidate-specific temporal score.

History and stability signals. KLASS combines confidence with a distribution-level KL signal (Kim et al., 2026). Anchor-based History-stable Decoding (AHD) uses historical anchors (Zou et al., 2026), and DC-Leap uses draft-guided contiguous verification and leaping (Jiao et al., 2026). LookUM and LOPA evaluate candidate continuations or local lookahead (Lee et al., 2025; Xu et al., 2025). TACG reads logits already produced by the backbone, scores the current candidate relative to an exponential-moving-average reference, and introduces no auxiliary forward pass for the gate.

Candidate-conditioned temporal contrast. TILG is algebraically analogous to classifier-free guidance because both form a contrastive readout from current and reference model outputs (Ho & Salimans, 2022). Classifier-free guidance combines conditional and unconditional score estimates to form a sampling distribution. TILG contrasts current logits with earlier logits from the same model

and uses the result to rank commitment candidates. The written token remains the base proposal, so TACG contributes a gate-level temporal readout alongside the base token distribution.

4 TEMPORAL HISTORY SELECTS COMMIT POSITIONS

4.1 THE BASE MODEL SUPPLIES EVERY WRITTEN TOKEN

Let $\mathcal{M}_t$ be the masked positions at step t, with logits $z_t(i,\cdot)$ and base posterior $p_t = \mathrm{softmax}(z_t)$. The base decoder supplies a proposal $\hat{x}_i$ and confidence $c_i = p_t(i,\hat{x}_i)$:

$$\hat{x}_i = \arg\max_v p_t(i,v), \qquad c_i = p_t(i,\hat{x}_i). \tag{3}$$

The gate decides whether to write this proposal. Figure 2 shows the information flow.

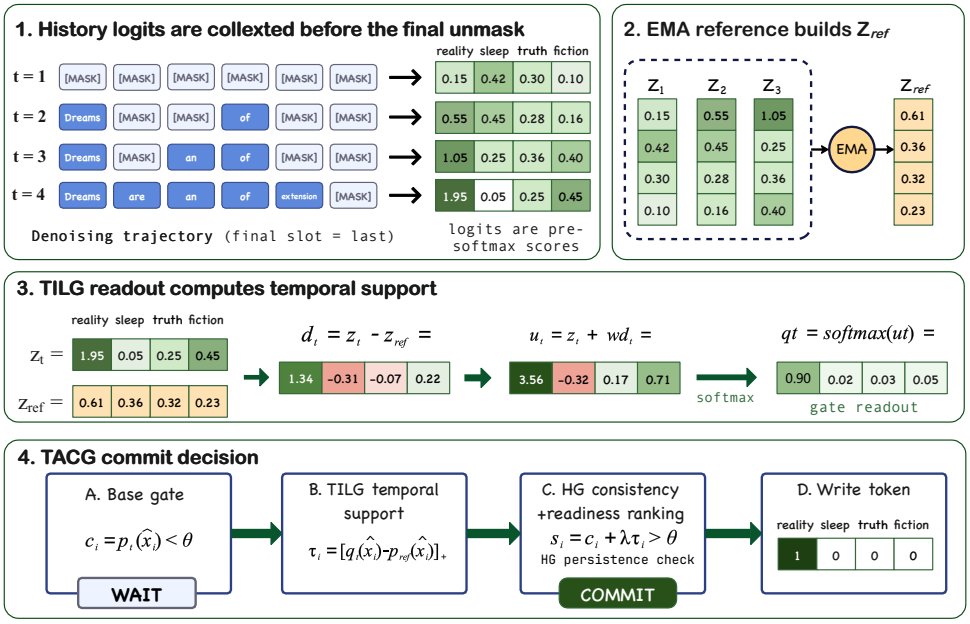


Figure 2: TACG uses a temporal-support branch to select commit positions. The base posterior supplies the current proposal. Earlier logits form an EMA reference, and TILG contrasts current logits with that reference to score candidate-conditioned support. History Gate and capped ranking select positions for commitment. Each write uses the current base proposal. The threshold sketch is illustrative; extra commitments follow the capped ranking rule in Section 4.3.

4.2 TILG SCORES CANDIDATE-CONDITIONED TEMPORAL SUPPORT

Current confidence describes a proposal's probability. Temporal support describes how the model's relative evidence for that proposal has changed. We maintain an EMA of earlier logits and read it before updating the reference:

$$\bar{z}_t = \beta \bar{z}_{t-1} + (1-\beta) z_t. \tag{4}$$

The first observation initializes the reference with current logits, so its support score is neutral. With $p_t^{\mathrm{ref}} = \mathrm{softmax}(\bar{z}_{t-1})$, the TILG readout is

$$q_t = \mathrm{softmax}\left(z_t + w(z_t - \bar{z}_{t-1})\right), \qquad q_t(v) \propto \frac{p_t(v)^{1+w}}{p_t^{\mathrm{ref}}(v)^w}. \tag{5}$$

For a pair of tokens, this readout adds w times the change in historical-to-current log odds to the current log odds. We query the readout at the base proposal and use its positive probability gain to

score the position:

$$b_i = \left[q_t(i, \hat{x}_i) - p_t^{\text{ref}}(i, \hat{x}_i)\right]_+, \qquad s_i = c_i + \lambda b_i. \tag{6}$$

The full vocabulary normalization matters, so a positive candidate log-ratio alone does not determine b_i. The parameters w and λ control the temporal contrast and its contribution to ranking. Appendix F gives the equivalent probability-ratio form and its first-order interpretation.

4.3 Persistence and a capped budget select positions

An abrupt support increase can occur before the proposed identity has persisted. The History Gate adds a discrete persistence condition to the continuous score. Let h_i count consecutive steps with the same proposal. For base and extra paths, define

$$H_i^j = \mathbf{1}[h_i \geq m_j] \vee \mathbf{1}[c_i \geq \tau_{\text{esc}}], \qquad j \in \{\text{base}, \text{extra}\}. \tag{7}$$

The confidence escape permits saturated predictions to pass immediately. A confidence floor defines the extra pool, and a per-step cap limits extra promotions. In the gate-before-ranking Dream adapter, the backend's base accept set $\mathcal{R}_t^{\text{base}}$ gives

$$\mathcal{B}_t = \{i \in \mathcal{R}_t^{\text{base}} : H_i^{\text{base}} = 1\}, \tag{8}$$

$$\mathcal{N}_t = \{i \in \mathcal{M}_t \setminus \mathcal{B}_t : c_i \geq \tau_{\text{floor}},\ H_i^{\text{extra}} = 1\}. \tag{9}$$

TACG promotes the highest-scoring candidates under the effective extra budget k_t:

$$\mathcal{E}_t = \text{TopK}_{i \in \mathcal{N}_t}(s_i, k_t), \qquad 0 \leq k_t \leq K_{\max}. \tag{10}$$

LLaDA preserves its backend ordering: confidence and KL eligibility define the base and extra pools, readiness ranking selects extras under a ratio-capped quota, and a shared History Gate filters their union. The quota uses the raw base-set size and the configured empty-base policy. Both adapters use a progress fallback and write the current base proposal at every accepted position. The cap bounds extra promotions, while the resulting trajectory determines quality and model-call cost. Appendix A gives both adapters explicitly, and Appendix B.1 lists the experimental settings.

4.4 The gate changes context timing while writes keep base identity

TACG computes its gate from logits already available at the current call, so gate computation adds no model forward pass. Its EMA and vocabulary operations scale as $O(BLV)$ for batch size B, active length L, and vocabulary size V. The write rule is local: every accepted position receives its contemporaneous base proposal. Earlier writes can change later contexts, so later proposals and final text can differ across commitment schedules. We therefore evaluate both model-call counts and measured decode latency, and we report logit-reference interventions as trajectory evidence with the local write rule stated explicitly. Appendix A gives the execution order for both adapters, including eligibility pools, quotas, fallback, state updates, and NFE counting; Tables 10 and 11 give the default parameters.

5 Experiments

5.1 Evaluation protocol anchors every quality–cost pair

We evaluate LLaDA-8B and Dream-7B on the complete HumanEval, MBPP, GSM8K, and MATH500 splits of 164, 257, 1,319, and 500 examples, respectively (Chen et al., 2021; Austin et al., 2021; Cobbe et al., 2021; Hendrycks et al., 2021). The main comparisons use one completion per example, seed 516, and generation length 256. We pair each method with its same-model baseline under the same task and generation budget. LLaDA uses temperature 0; Dream uses temperature 0.2 and top-p 0.95.

We report task accuracy (ACC) and mean main-model forward evaluations (NFE) per example. Lower NFE indicates fewer model calls. Each accuracy–NFE pair comes from one complete-task run, and aggregate values average the four benchmarks equally. LLaDA2-Mini uses separate protocols at generation lengths 1024 and 2048 with MBPP denominator 427. Full protocol details and adapted baseline schedules appear in Appendix B. The records in `analysis/revision_table_data.json`,

Table 1: Accuracy and model-call cost on LLaDA-8B. ACC is accuracy (%); NFE is mean model calls per example. Mean averages the four tasks equally; bold marks column best. Shaded rows mark TACG; unshaded rows are baselines or controls.

Method	HumanEval		MBPP		GSM8K		MATH500		Mean	
	ACC ↑	NFE ↓	ACC ↑	NFE ↓	ACC ↑	NFE ↓	ACC ↑	NFE ↓	ACC ↑	NFE ↓
Native Confidence	37.20	52.28	47.08	83.34	77.10	74.26	30.60	96.48	48.00	76.59
AHD	39.63	51.89	45.91	85.46	75.28	73.97	30.40	95.28	47.81	76.65
DSB	39.02	50.46	48.64	81.49	76.35	72.98	33.40	95.68	49.35	75.15
KLASS	39.63	92.13	48.25	134.82	**77.63**	134.45	30.60	152.42	49.03	128.45
TACG	**42.07**	**45.71**	**50.97**	**74.42**	77.03	**67.12**	**34.00**	**84.04**	**51.02**	**67.82**

`analysis/supplementary_data/candidate_results.csv`, and the audited latency/intervention tables retain task denominators, configuration vectors, and source-path checks.

Native Confidence commits positions with the backend's current-confidence rule and serves as the same-model baseline. The headline TACG rows use benchmark-specific operating points. We report shared vectors and independently timed vectors as separate settings.

5.2 Main quality–cost comparison

Table 1 gives the headline LLaDA comparison. The displayed task-specific TACG points are measured on completed full-task runs from the candidate roster under the stated single-seed protocol. They improve accuracy on HumanEval, MBPP, and MATH500 and reduce model calls on all four benchmarks. Mean accuracy rises from 48.00% to 51.02%, and mean NFE falls from 76.59 to 67.82. Among the displayed rows, TACG has the highest mean accuracy. The fixed KLASS row provides an additional GSM8K operating point. The component and replay studies below test whether the quality–cost change follows from which positions the decoder commits.

5.3 Selection and writing contribute differently in a two-by-two intervention

Table 2: Full task results for the component diagnostic. Mean averages the four complete tasks; Selection is history-guided position choice and Writing changes token values.

Method	Metric	HumanEval	MBPP	GSM8K	MATH500	Mean
Confidence	ACC ↑	37.20	47.08	77.10	30.60	48.00
	NFE ↓	52.28	83.34	74.26	96.48	76.59
Selection	ACC ↑	39.02	50.19	76.04	32.60	49.47 (+1.47)
	NFE ↓	48.02	74.52	63.83	85.09	67.87 (-11.39%)
Writing	ACC ↑	37.80	49.03	75.36	31.80	48.50 (+0.50)
	NFE ↓	50.65	82.93	74.74	97.13	76.36 (-0.30%)
Selection + writing	ACC ↑	40.24	49.81	75.51	32.60	49.54 (+1.55)
	NFE ↓	46.16	74.00	64.16	85.80	67.53 (-11.83%)

The component study in Table 2 uses an independent across-task LLaDA batch, so its absolute values differ from the headline task-specific rows. The history-guided selection policy is composite: it combines temporal ranking, capped extra promotion, persistence and escape checks, and progress fallback. Relative to its matched control, the selection arm changes mean accuracy by +1.47 points and mean NFE by -11.39%; the guided-token-writing arm changes accuracy by +0.50 points and NFE by -0.30%. The combined variant reaches 49.54% accuracy with 67.53 mean NFE. The diagnostic shows a larger observed shift in the composite selection arm under this intervention. The factorization supports placing temporal evidence in the selector while the base decoder supplies the written token.

Table 3: Matched per-event commit-count selector replay. Both selectors use identical per-sample NFE; selected positions differ.

Selector	ACC ↑	NFE ↓
History-guided	49.42	62.79
Confidence-ranked	41.16	62.79

Table 4: Per-task results under matched per-event commit counts.

Selector	Metric	HumanEval	MBPP	GSM8K	MATH500	Mean
History-guided	ACC ↑	39.02	50.19	76.27	32.20	49.42
	NFE ↓	44.02	69.39	58.33	79.43	62.79
Confidence-ranked	ACC ↑	34.76	38.52	67.55	23.80	41.16
	NFE ↓	44.02	69.39	58.33	79.43	62.79

5.4 History selection matters under matched per-event commit counts

The replay tests position choice under matched per-event commit counts. The confidence-ranked control reproduces the reference's total commit count at every sample, block, and event, so both selectors use exactly the same per-sample NFE. History-guided selection reaches 49.42% accuracy versus 41.16% for confidence ranking, a difference of 8.27 percentage points across the four tasks.

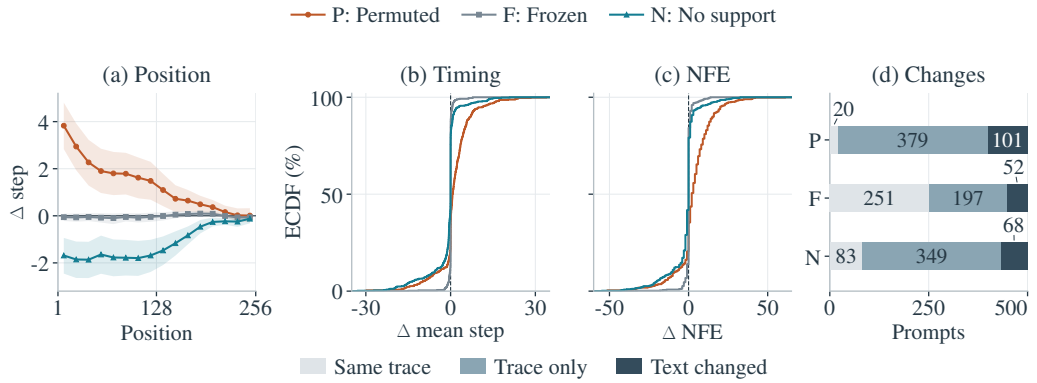


Figure 3: Dream MATH500 intervention diagnostic on 500 paired prompts. (a) Mean commit-step changes across 16-position bins with pointwise 95% intervals; (b,c) ECDFs of prompt-level mean commit-step and NFE changes; (d) joint changes in position-wise (step, source) traces and final text.

5.5 Logit-reference interventions change trajectories while local writes stay anchored

The four arms use the observed EMA reference, a frozen reference, no temporal support, and a within-sample position-permuted reference. The no-support arm removes the temporal-support branch; the other intervention arms alter its reference. Confidence and the proposal-streak gate remain fixed. Relative to the observed reference, the three interventions change 417, 249, and 480 of 500 commit traces, and 68, 52, and 101 final texts, respectively. All 512,000 recorded writes match their local base proposal. All three controls have paired accuracy intervals; together, these controls provide a trajectory-sensitivity view of the commit path and final text.

5.6 The measured roster covers quality–NFE operating points

Figure 4 shows the measured LLaDA quality–NFE roster. Each orange point is a complete-task evaluation and the line joins the measured non-dominated points. The main table selects one operating point per benchmark; the appendix reports the complete LLaDA and Dream candidate rosters.

The sensitivity panels vary one control at a time on LLaDA HumanEval. The support-weight slice covers $\lambda = 0.15$–0.75 at $w = 0.35$; the headline HumanEval operating point uses $\lambda = 2.00$ and is outside this slice. The prompt-level panel evaluates six History Gate settings on the same 164 examples.

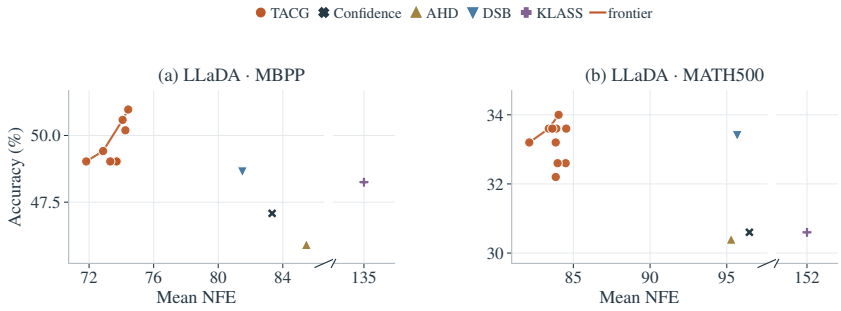


Figure 4: **Measured quality–NFE operating points on LLaDA.** We compare TACG settings with Native Confidence, AHD, DSB, and KLASS. The line connects the measured non-dominated TACG points. The appendix reports the controlled λ slice and Dream sweep.

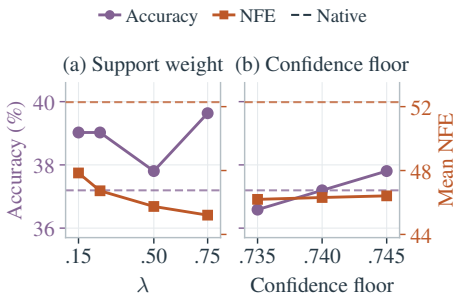


Figure 5: **Single-parameter sensitivity on LLaDA HumanEval**. We vary λ from 0.15 to 0.75 at $w = 0.35$ and the confidence floor from 0.735 to 0.745, one parameter at a time. Dashed lines denote Native Confidence; these slices are separate from the main HumanEval operating point.

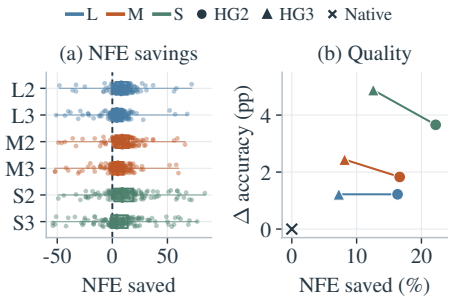


Figure 6: Prompt-level call changes on HumanEval across six History Gate settings (984 paired records). L/M/S label the low, moderate, and strong support vectors in Table 7; 2/3 denote the persistence windows. (a) NFE saved is Native minus TACG. (b) Mean relative NFE saving and accuracy change.

Table 5: Accuracy and model-call cost across backbones. Mean averages the four tasks equally; Dream uses MBPP257 and LLaDA2-Mini uses MBPP427. Shaded rows mark TACG.

Method	HumanEval		MBPP		GSM8K		MATH500		Mean	
	ACC ↑	NFE ↓	ACC ↑	NFE ↓	ACC ↑	NFE ↓	ACC ↑	NFE ↓	ACC ↑	NFE ↓
LLaDA-8B										
Native Confidence	37.20	52.28	47.08	83.34	**77.10**	74.26	30.60	96.48	48.00	76.59
TACG	**42.07**	**45.71**	**50.97**	**74.42**	77.03	**67.12**	**34.00**	**84.04**	**51.02**	**67.82**
Dream-7B										
Native Confidence	47.56	51.19	**63.42**	**82.46**	73.16	74.63	39.00	94.60	55.79	**75.72**
TACG	**50.00**	**50.80**	63.04	93.53	**73.62**	**74.26**	**39.60**	**91.14**	**56.56**	77.43
LLaDA2-Mini										
Baseline	84.15	**67.38**	80.09	77.93	87.64	**147.90**	64.80	294.77	79.17	146.99
TACG	**85.98**	69.66	**80.56**	**75.36**	**88.17**	148.11	**67.40**	**290.79**	**80.53**	**145.98**

5.7 Adapter-specific tests span backbones and lengths

Table 5 applies the same temporal-support framework, adapted to three decoder families. On the tested Dream task-specific operating points, TACG improves both accuracy and call count on HumanEval, GSM8K, and MATH500; MBPP supplies the complementary operating point under the same official adapter. On LLaDA2-Mini, accuracy improves on all four tasks at both 1024 and 2048 generated

Table 6: LLaDA2-Mini at two generation budgets. Each panel matches the generation budget; MBPP uses 427 examples and shaded rows mark TACG.

Method	HumanEval		MBPP		GSM8K		MATH500		Mean	
	ACC ↑	NFE ↓	ACC ↑	NFE ↓	ACC ↑	NFE ↓	ACC ↑	NFE ↓	ACC ↑	NFE ↓
Generation length 1024										
Baseline	84.15	**67.38**	80.09	77.93	87.64	**147.90**	64.80	294.77	79.17	146.99
TACG	**85.98**	69.66	**80.56**	**75.36**	**88.17**	148.11	**67.40**	**290.79**	**80.53**	**145.98**
Generation length 2048										
Baseline	84.15	**71.51**	80.09	**78.12**	88.32	176.97	75.20	456.83	81.94	195.86
TACG	**85.37**	78.32	**80.33**	79.72	**88.63**	**171.18**	**76.40**	**451.52**	**82.68**	**195.19**

Table 7: History Gate window on HumanEval. Each pair changes only the minimum proposal streak; the three rows use distinct support settings.

	HG = 2		HG = 3	
(w, λ)	ACC	NFE	ACC	NFE
(0.30, 0.05)	**38.41**	**43.75**	**38.41**	48.49
(0.35, 0.10)	39.02	**43.58**	**39.63**	48.02
(0.38, 2.00)	40.85	**40.68**	**42.07**	45.71

Table 8: Matched request cost at two HG windows on H200. Each setting uses the same 80 timing requests.

Metric	HG = 2	HG = 3
NFE	**85.05**	90.10
Decode (ms)	**2145.88**	2279.48
Request (ms)	**2334.90**	2427.97
Wall speedup	**1.040**×	1.000×

positions, with mean NFE decreasing from 146.99 to 145.98 and from 195.86 to 195.19. The table below lists the matched task-level results; the two generation budgets remain separate because they correspond to different block trajectories.

5.8 Measured decode cost is separate from NFE

The default rows use benchmark-specific points; the appendix reports one shared vector and a separate timing-study vector. We measure request-level decoding behavior instead of inferring wall-clock gains from NFE. The paired History Gate study changes only the persistence window, and the latency table uses matched requests under its own configuration. The fully profiled timing-study vector

Table 9: Matched decoding latency for alternate LLaDA parameter settings. Mean synchronized decode time (ms) on H200; speedup is the four-task mean ratio.

Method	Setting	HumanEval	MBPP	GSM8K	MATH500	Mean ms ↓	Speedup ↑
Native Confidence	Reference	1391.3	2073.7	1975.5	2341.0	1945.4	1.000×
TACG	Across-task	1330.5	1949.8	1784.9	2119.7	1796.2	1.083×
TACG	Timing study	1383.0	1780.1	1836.4	2119.7	1779.8	1.093×

gives a 1.093× decode-speed ratio on LLaDA, with mean synchronized decode times of 1779.8 ms and 1945.4 ms. The timing-study vector is a separate fully profiled quality–NFE point with 50.47% quality and 66.82 mean NFE. Appendix E.2 reports throughput and memory on the tested length–batch grid.

6 Conclusion

TACG uses the denoising trajectory to decide when a base proposal should enter the context. Its temporal-support and persistence signals change position selection while retaining the current proposal at every write. Complete-task comparisons on the tested single-seed LLaDA roster show accuracy gains on three of four benchmarks with fewer calls on all four, and controlled interventions show observed selector differences under matched per-event commit counts and identical per-sample NFE. Dream, LLaDA2-Mini, and the system profile characterize tested operating points across backbones, lengths, and throughput configurations.

REFERENCES

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

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces, 2023. URL https://arxiv.org/abs/2107.03006.

Tiwei Bie, Maosong Cao, Kun Chen, Lun Du, Mingliang Gong, Zhuochen Gong, Yanmei Gu, Jiaqi Hu, Zenan Huang, Zhenzhong Lan, Chengxi Li, Chongxuan Li, Jianguo Li, Zehuan Li, Huabin Liu, Lin Liu, Guoshan Lu, Xiaocheng Lu, Yuxin Ma, Jianfeng Tan, Lanning Wei, Ji-Rong Wen, Yipeng Xing, Xiaolu Zhang, Junbo Zhao, Da Zheng, Jun Zhou, Junlin Zhou, Zhanchao Zhou, Liwang Zhu, and Yihong Zhuang. Llada2.0: Scaling up diffusion language models to 100b, 2025. URL https://arxiv.org/abs/2512.15745.

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

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

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, 2021. URL https://arxiv.org/abs/2103.03874.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance, 2022. URL https://arxiv.org/abs/2207.12598.

Daniel Israel, Guy Van den Broeck, and Aditya Grover. Accelerating diffusion llms via adaptive parallel decoding, 2025. URL https://arxiv.org/abs/2506.00413.

Yanhua Jiao, Tianyi Wu, Xiaoxi Sun, Yulin Li, Huiling Zhen, Libo Qin, Baotian Hu, Zhuotao Tian, and Min Zhang. DC-Leap: Training-free acceleration of dLLMs via draft-guided contiguous leaping decoding, 2026. URL https://arxiv.org/abs/2607.20467.

Seo Hyun Kim, Sunwoo Hong, Hojung Jung, Youngrok Park, and Se-Young Yun. Klass: Kl-guided fast inference in masked diffusion models, 2026. URL https://arxiv.org/abs/2511.05664.

Fanheng Kong, Jingyuan Zhang, Yahui Liu, Zirui Wu, Yu Tian, Victoria W., and Guorui Zhou. Accelerating diffusion llm inference via local determinism propagation, 2025. URL https://arxiv.org/abs/2510.07081.

Sanghyun Lee, Seungryong Kim, Jongho Park, and Dongmin Park. Lookahead unmasking elicits accurate decoding in diffusion language models, 2025. URL https://arxiv.org/abs/2511.05563.

Pengxiang Li, Yefan Zhou, Dilxat Muhtar, Lu Yin, Shilin Yan, Li Shen, Soroush Vosoughi, and Shiwei Liu. Diffusion language models know the answer before decoding, 2026. URL https://arxiv.org/abs/2508.19982.

Lizhuo Luo, Shenggui Li, Yonggang Wen, and Tianwei Zhang. DSB: Dynamic sliding block scheduling for diffusion LLMs, 2026. URL https://arxiv.org/abs/2602.05992.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models, 2025. URL https://arxiv.org/abs/2502.09992.

Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models, 2024. URL https://arxiv.org/abs/2406.07524.

Chengyue Wu, Hao Zhang, Shuchen Xue, Zhijian Liu, Shizhe Diao, Ligeng Zhu, Ping Luo, Song Han, and Enze Xie. Fast-dllm: Training-free acceleration of diffusion llm by enabling kv cache and parallel decoding, 2025. URL https://arxiv.org/abs/2505.22618.

Chenkai Xu, Yijie Jin, Jiajun Li, Yi Tu, Guoping Long, Dandan Tu, Mingcong Song, Hongjie Si, Tianqi Hou, Junchi Yan, and Zhijie Deng. Lopa: Scaling dllm inference via lookahead parallel decoding, 2025. URL https://arxiv.org/abs/2512.16229.

Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Dream 7b: Diffusion large language models, 2025. URL https://arxiv.org/abs/2508.15487.

Shun Zou, Yong Wang, Zehui Chen, Lin Chen, Chongyang Tao, Feng Zhao, and Xiangxiang Chu. Breaking block boundaries: Anchor-based history-stable decoding for diffusion large language models, 2026. URL https://arxiv.org/abs/2604.08964.

A DECODING ALGORITHM

The listings give the shared TACG computations and the LLaDA and Dream selection rules used in our experiments. Each listing targets one sequence; we apply selection independently to each active batch row. `topk(score, candidates, k)` returns a Boolean mask containing the highest-scoring $\min(k, |\text{candidates}|)$ eligible positions, and returns an empty mask when $k = 0$.

Shared temporal state and readout. The state contains the EMA logits, the previous base proposal, and its consecutive-agreement count. The first observation gives zero temporal support. The historical reference is read before the current logits enter the EMA.

```
class TACGState:
    def __init__(self):
        self.ref = None
        self.previous = None
        self.streak = None


def tacg_readout(logits, history_mask, state, cfg):
    p = softmax(logits, dim=-1)
    proposal = argmax(p, dim=-1)
    confidence = gather(p, proposal)
    if state.previous is None:
        streak = ones_like(proposal)
    else:
        same = proposal == state.previous
        streak = where(same, state.streak + 1, 1)
    streak = where(history_mask, streak, 0)
    ref = logits if state.ref is None else state.ref
    q = softmax(logits + cfg.w * (logits - ref), dim=-1)
    p_ref = softmax(ref, dim=-1)
    support = clamp(gather(q, proposal)
                    - gather(p_ref, proposal), min=0)
    if state.ref is None:
        support = zeros_like(confidence)
    score = confidence + cfg.support_lambda * support
    return p, proposal, confidence, streak, score, ref


def update_state(state, logits, proposal, streak, ref, beta):
    state.ref = detach(beta * ref + (1 - beta) * logits)
    state.previous = detach(proposal)
    state.streak = detach(streak)
```

Base eligibility and backend selection. Let `kl_ok` be true for a confidence-only gate. For an active KLASS gate it requires the backend's configured window of consecutive values $D_{\mathrm{KL}}(p_t \| p_{t-1})$ to fall below its KL threshold, using the same zero-initialized posterior/window and numerical offset as the backend. LLaDA retains this eligibility check for both raw base candidates and extra candidates. Its ratio cap uses the number of raw base candidates, and its History Gate filters the combined candidate set after extra selection. Dream applies separate persistence thresholds to the base and extra paths and uses a fixed extra cap.

```
def llada_select(mask, confidence, streak, score, kl_ok,
                 scheduled_k, final_step, cfg):
    raw_base = mask & kl_ok & (confidence > cfg.tau_conf)
    eligible = (mask & kl_ok
                & (confidence > cfg.tau_floor)
                & (confidence <= cfg.tau_conf))
    k = min(cfg.K_max, ceil(cfg.extra_ratio
                            * max(count(raw_base), 1)))
    if not any(raw_base) and not cfg.allow_empty_base:
        k = 0
    extra = topk(score, eligible, k)
    candidates = raw_base | extra
    if not any(candidates):
        candidates = topk(score, mask, scheduled_k)
    history_ok = ((streak >= cfg.m_base)
                  | (confidence >= cfg.tau_escape))
    commit = candidates & history_ok & mask
    if not any(commit) and any(candidates):
        commit = topk(score, candidates, 1)
    elif not any(commit) and final_step:
        commit = topk(score, mask, scheduled_k)
    return commit


def dream_select(logits, mask, confidence, streak, score,
                 kl_ok, cfg):
    raw_base = mask & kl_ok & (confidence >= cfg.tau_conf)
    escape = confidence >= cfg.tau_escape
    base = raw_base & ((streak >= cfg.m_base) | escape)
    eligible = (mask & ~base
                & (confidence >= cfg.tau_floor)
                & ((streak >= cfg.m_extra) | escape))
    extra = topk(score, eligible, cfg.K_max)
    commit = base | extra
    if not any(commit) and any(mask):
        # Same position selection as the Dream baseline.
        fallback_conf, _ = sample_tokens(
            logits[mask], temperature=cfg.temperature,
            top_p=cfg.top_p, top_k=cfg.top_k)
        fallback_score = full_like(confidence, -inf)
        fallback_score[mask] = fallback_conf
        commit = topk(fallback_score, mask, 1)
    return commit
```

The Dream fallback samples from the backend's temperature-scaled and truncated posterior to score positions; we discard the sampled token. The write below always uses the current base-posterior argmax. In LLaDA, the same persistence threshold applies to base and extra candidates, and HG rejects an extra candidate without replacing it with a lower-ranked candidate. We account for mandatory progress separately from the extra cap.

Block loop, writes, and model-call counting. LLaDA uses its configured blocks, retains temporal state across blocks, and tracks proposal streaks over the full sequence. Dream treats the generation region as one active block and updates streaks at masked positions. For a block with n initially masked positions and S scheduled steps, LLaDA assigns $\lfloor n/S \rfloor + \mathbf{1}[t < n \bmod S]$ fallback positions to step t. The following loop shows where these state updates and the NFE counter occur; the loop computes `kl_ok` from the same forward result. The reported confidence settings keep the KL check open; active KLASS configurations retain the window check shown below.

```
x = concatenate(prompt, full(generation_length, MASK))
state, nfe = TACGState(), 0
previous_p, kl_history = None, zeros(len(x), cfg.kl_window)
for block in generation_blocks:
    initial_count = count((x == MASK) & block)
    steps = cfg.steps_for_block(block)
    for t in range(steps):
        mask = (x == MASK) & block
        if not any(mask):
            break
        logits = model(x).logits
        nfe += 1                              # one actual model call
        logits = align_backend_logits(logits)
        history_mask = ones_like(mask) if cfg.llada else mask
        p, proposal, conf, streak, score, ref = tacg_readout(
            logits, history_mask, state, cfg)
        kl_ok = ones_like(mask)
        if cfg.use_kl:
            if previous_p is None:
                previous_p = zeros_like(p)
            d = sum(p * (log(p + 1e-12)
                         - log(previous_p + 1e-12)), dim=-1)
            kl_history = append_and_drop_oldest(kl_history, d)
            kl_ok = all(kl_history < cfg.tau_kl, dim=-1)
            if cfg.llada and t < cfg.kl_window - 1:
                kl_ok = zeros_like(mask)
            previous_p = detach(p)
        if cfg.llada:
            budget = (initial_count // steps
                      + (t < initial_count % steps))
            commit = llada_select(mask, conf, streak, score,
                                  kl_ok, budget, t == steps - 1, cfg)
        else:
            commit = dream_select(logits, mask, conf, streak,
                                  score, kl_ok, cfg)
        x[commit] = proposal[commit]       # base, extra, and fallback
        update_state(state, logits, proposal, streak, ref, cfg.beta)
        mask = mask & ~commit
return x, nfe
```

`align_backend_logits` applies the backbone's output-position convention (Dream shifts its logits by one position). TACG's readout and selection need no additional model evaluation. The LLaDA2-Mini implementation retains its block and cache schedule; its reported decoding NFE also includes cache-refresh model calls and excludes prompt prefill. Experimental settings appear in Appendix B.1.

B Protocols and Configurations

Complete task evaluation. We use generation length 256 for the main LLaDA-8B/Dream-7B tables and ordered, complete task splits with one generated completion per example. We evaluate LLaDA2-Mini separately at lengths 1024 and 2048; baseline and TACG share the generation budget within each comparison. We score code-task accuracy on the saved completions and compute NFE from per-example model-call counters. Accuracy, mean NFE, and latency have distinct denominators:

the first two cover complete tasks, whereas matched decode profiles use repeated timing requests. We average accuracy and NFE across the four tasks within each panel.

On GSM8K, the LLaDA Native Confidence and TACG rows use the same 1,319 test questions, checkpoint, prompts, and answer parser, with generation length 256, block length 64, and step budget 256. Their saved per-question results give 1,017 and 1,016 correct answers and mean NFE 74.26 and 67.12, respectively. We aggregate both metrics over the complete test split.

Adapted baselines. Native Confidence and fixed KLASS serve as separate gate families; KLASS uses confidence threshold 0.9, KL threshold 0.01, and history length 2 on all tasks. AHD and DSB use the authors' implementations adapted to the evaluation protocol with their method-specific schedules. AHD uses a fixed KL threshold of 0.01 across tasks; DSB uses its greedy scheduler with caching disabled. NFE counts main-model evaluations, whose computational cost can differ across implementations.

Operating-point selection. The default TACG row uses benchmark-specific operating points. The across-task ablation uses one TACG vector for all tasks within each model.

B.1 Hyperparameter settings

The following tables list the registered vectors and full candidate rosters. Both backbones use EMA decay $\beta = 0.95$. LLaDA uses HG streak 3, at most two extra commits, extra ratio 0.25, and empty-base promotion. For Dream, m_b and m_e denote the base and extra-path streak requirements, and K is the maximum extra-commit budget. The main Dream row uses one benchmark-specific accuracy–NFE operating point per task with the same current TACG adapter; the accompanying machine-readable records retain the corresponding raw paths and receipts. Tables 12 and 13 give

Table 10: TACG parameters on LLaDA. The main comparison uses Default; the Timing study setting uses Default vectors for HumanEval/GSM8K and the separate vectors below for MBPP/MATH500. Across-task uses one vector for all tasks. All use confidence threshold 0.9, EMA decay 0.95, at most two extra commits, and extra ratio 0.25. The settings use empty-base promotion.

Setting	Task	w	λ	τ_{floor}	HG	τ_{esc}
Across-task	All tasks	0.35	0.1	0.74	3	0.95
Default	HumanEval	0.38	2	0.74	3	0.96
Default	MBPP	0.35	0.15	0.74	3	0.95
Default	GSM8K	0.3	0.05	0.8	3	0.95
Default	MATH500	0.365	0.35	0.745	3	0.95
Timing study	MBPP	0.35	0.1	0.74	2	0.95
Timing study	MATH500	0.35	0.1	0.74	3	0.95

Table 11: TACG parameters on Dream. Default gives the benchmark-specific operating points used in the main comparison; Across-task uses one vector for all tasks. All rows use the same TACG adapter with EMA decay 0.95. Base confidence is 0.95 on MBPP and 0.90 on the other tasks; m_b and m_e give the base and extra-path streaks, and K caps extra commits.

Setting	Task	w	λ	τ_{floor}	m_b	m_e	τ_{esc}	K
Across-task	All tasks	0.3	0.02	0.885	5	3	0.99	1
Default	HumanEval	0.3	0.02	0.895	1	1	1	1
Default	MBPP	0.3	0.02	0.89	5	3	0.99	2
Default	GSM8K	0.225	0.015	0.899	1	1	1	1
Default	MATH500	0.35	0.1	0.88	3	1	0.95	2

the complete candidate vectors used by the operating-point analysis.

Table 12: LLaDA sweep: each row is one parameter vector, applied to all four tasks.

w	λ	τ_{floor}	τ_{esc}
0.35	0.15	0.74	0.95
0.35	0.25	0.74	0.95
0.365	0.25	0.74	0.95
0.365	0.35	0.74	0.95
0.365	0.35	0.745	0.95
0.365	0.35	0.74	0.96
0.38	0.35	0.74	0.95
0.35	0.5	0.74	0.95
0.365	0.35	0.735	0.95
0.35	0.75	0.74	0.95

Table 13: Dream sweep: each row is one parameter vector, applied to all four tasks.

w	λ	τ_{floor}	m_b	m_e	τ_{esc}	K
0.3	0.02	0.885	5	3	0.99	1
0.3	0.02	0.885	4	3	0.99	1
0.3	0.02	0.88	4	3	0.99	1
0.3	0.02	0.88	5	2	0.99	1
0.3	0.02	0.875	5	3	0.99	1
0.3	0.02	0.88	5	3	0.985	1
0.3	0.035	0.88	5	3	0.99	1
0.275	0.02	0.88	5	3	0.99	1
0.225	0.015	0.89	5	3	0.99	1
0.3	0.02	0.89	5	3	0.99	2

C Complete candidate results

C.1 LLaDA2-Mini at longer generation lengths

Table 6 reports both Mini generation budgets, including the 1024-token setting used in Table 5. The comparisons use complete HumanEval, MBPP, GSM8K, and MATH500 splits with 164, 427, 1,319, and 500 examples. Mini uses the historical MBPP427 protocol, separate from the MBPP257 split used for LLaDA-8B and Dream; NFE counts decoding calls across blocks and cache refreshes, excluding prompt prefill. We evaluate both lengths with task-specific TACG settings, with the 2048-token History Gate enabled for HumanEval and MBPP. The two generation budgets are separate operating points.

C.2 Complete quality–NFE candidate coverage

We evaluate the ten declared configurations per backbone on all four complete tasks, yielding 80 candidate task results and 44,800 generated samples. Each configuration shares its TACG vector across tasks within its model, with the audited task-specific base-gate presets retained. Figure 7 gives the complete two-backbone candidate view; the parameter tables and machine-readable candidate table retain the corresponding vectors and task results. The default rows of Tables 1 and 5 use the benchmark-specific vectors of Tables 10 and 11. The LLaDA MBPP and MATH500 defaults and the Dream MBPP default coincide with sweep members; the remaining defaults are separate complete-task runs with the same adapters.

D Mechanism audits and sensitivity

D.1 Selection and token-writing interventions

The selection intervention enables the shared history-guided position policy: temporal-bonus ranking, capped extra promotion, proposal-streak/escape checks, and progress fallback. The writing intervention changes the written token to the guided-logit argmax. Their four-way comparison shares inputs and saved configuration fields apart from the algorithm switch. The study uses matched runs for each intervention.

D.2 Replayed per-event commit counts

The history-guided reference run supplies 141,707 commit events per arm. The confidence-ranked control reproduces the total committed-token count at every aligned sample/block/event and therefore the exact per-sample NFE. Confidence ranking selects current top-confidence positions; the detailed table retains the same two selectors with their task-level values.

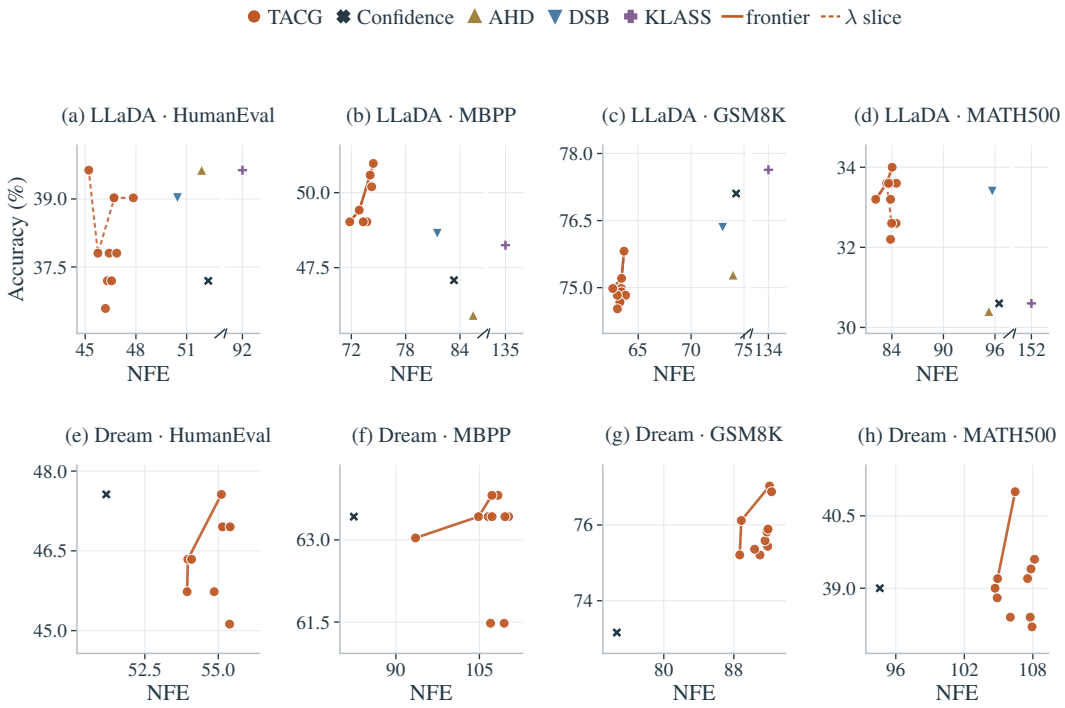


Figure 7: Complete two-backbone candidate sweep: 80 TACG task results across four benchmarks and the available baseline overlays. Each panel retains all ten shared-vector candidates; the LLaDA panels include Native Confidence, AHD, DSB, and the fixed-vector KLASS control, while Dream includes its audited Native Confidence overlay. The solid orange line joins each panel's non-dominated candidates, as in Figure 4. The dashed orange segments in the LLaDA panels connect the controlled four-point λ slice, $\lambda \in \{0.15, 0.25, 0.50, 0.75\}$ with all remaining settings fixed, in parameter order; the two lines coincide on MBPP and differ elsewhere. Table 5 reports the current Dream task-specific rows; the main-text LLaDA panels show selected views of this complete sweep.

Table 14: Full task results for fixed-commit-budget replay. Both selectors replay identical per-event commit counts, so their per-sample NFE matches.

Method	Metric	HumanEval	MBPP	GSM8K	MATH500	Mean
History-guided	ACC ↑	39.02	50.19	76.27	32.20	49.42
	NFE ↓	44.02	69.39	58.33	79.43	62.79
Confidence-ranked	ACC ↑	34.76	38.52	67.55	23.80	41.16
	NFE ↓	44.02	69.39	58.33	79.43	62.79

D.3 PAIRED LOGIT-REFERENCE INTERVENTIONS

The Dream diagnostic uses the same 500 MATH500 prompts across four arms at seed 516, temperature 0.2, top-p 0.95, and generation length 256. The four arms use the observed EMA reference, a frozen reference, no temporal support, and a within-sample position-permuted reference. Confidence and the proposal-streak gate remain fixed. Position permutation acts across positions, not across time. The position curve aggregates all 256 generated slots into 16-position bins, with paired prompt bootstrap intervals. The empirical distribution plots retain the full prompt population and both tails. A trace change means a difference in a position's (commit step, commit source) tuple; a text change means a difference in the final generated string. All recorded writes across the four arms match their local proposals.

D.4 SENSITIVITY SETTINGS AND HISTORY GATE WINDOWS

Figure 5 evaluates seven settings on the same 164 HumanEval problems at generation length 256. The support-weight slice varies λ with $w = 0.35$ and $\tau_{\text{floor}} = 0.74$; the confidence-floor slice varies τ_{floor} with $w = 0.365$ and $\lambda = 0.35$. Only the horizontal-axis parameter changes within each slice. Both use HG streak 3 and EMA decay 0.95. Each point pairs accuracy and NFE from the same complete run.

The sensitivity plot uses the same LLaDA HumanEval protocol as the main table and varies one control at a time. The system profiles below use matched requests to characterize the cost of different History Gate windows.

E SHARED SETTINGS AND SYSTEM PROFILES

E.1 CROSS-TASK SETTINGS

Table 15 compares the benchmark-specific operating points with one vector shared across tasks and with the separate timing-study vector. The main text uses the benchmark-specific rows for the headline quality–NFE comparison.

Table 15: Effect of across-task and per-task parameter settings. Reference is Native Confidence; Mean averages the four complete tasks.

Method	Setting	Metric	HumanEval	MBPP	GSM8K	MATH500	Mean
LLaDA-8B							
Native Confidence	Reference	ACC ↑	37.20	47.08	77.10	30.60	48.00
		NFE ↓	52.28	83.34	74.26	96.48	76.59
TACG	Across-task	ACC ↑	39.63	50.19	76.04	32.60	49.62
		NFE ↓	48.02	74.52	63.83	85.09	67.87
TACG	Timing study	ACC ↑	42.07	50.19	77.03	32.60	50.47
		NFE ↓	45.71	69.35	67.12	85.09	66.82
TACG	Default	ACC ↑	42.07	50.97	77.03	34.00	51.02
		NFE ↓	45.71	74.42	67.12	84.04	67.82
Dream-7B							
Native Confidence	Reference	ACC ↑	47.56	63.42	73.16	39.00	55.79
		NFE ↓	51.19	82.46	74.63	94.60	75.72
TACG	Across-task	ACC ↑	46.95	63.81	77.03	39.20	56.75
		NFE ↓	55.32	108.33	92.08	107.54	90.82
TACG	Default	ACC ↑	50.00	63.04	73.62	39.60	56.56
		NFE ↓	50.80	93.53	74.26	91.14	77.43

E.2 Length and batch scaling

The scaling study uses a fixed 66-token prefix replicated across each batch on one H200 GPU (bfloat16, seed 516). It contains 16 conditions: four lengths for the original decoder at batch 1, and the same lengths for shared TACG at batches 1, 4, and 8. Each condition has five warm-ups, 20 timing repeats, and five separate memory repeats. The original decoder uses block length 32 and TACG uses 64. These profiles characterize throughput, latency, model-call count, and memory on the tested grid.

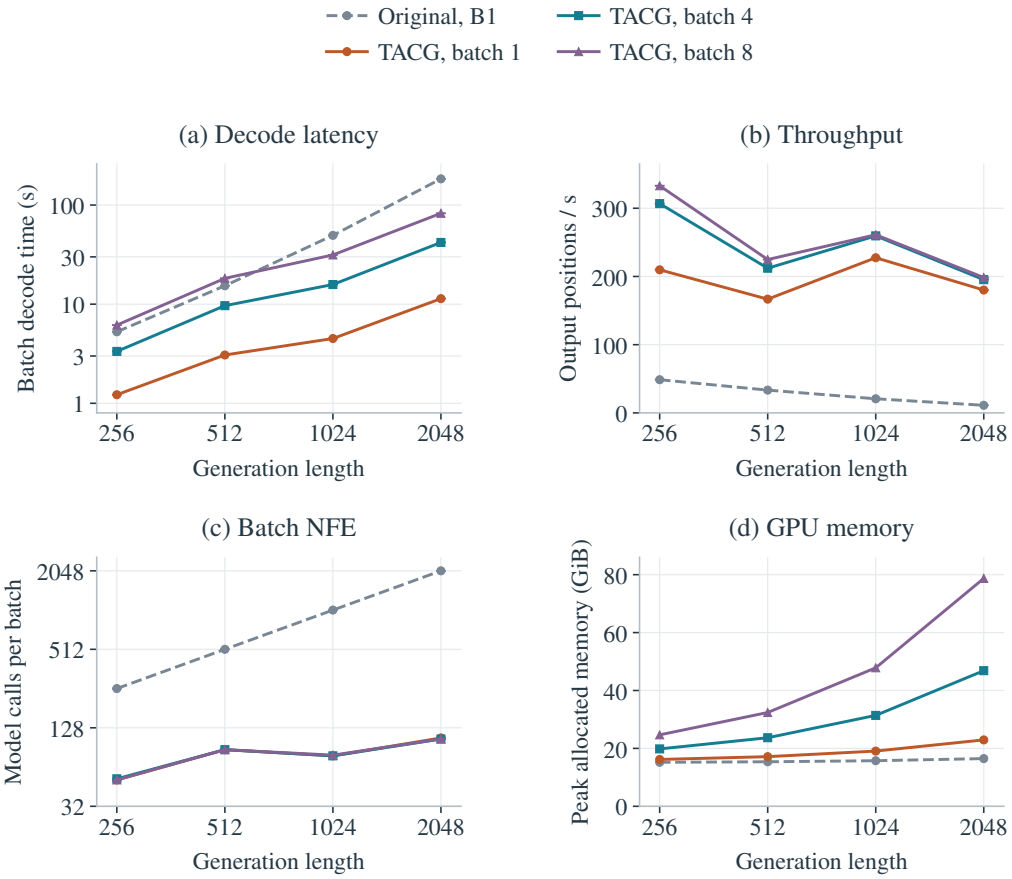


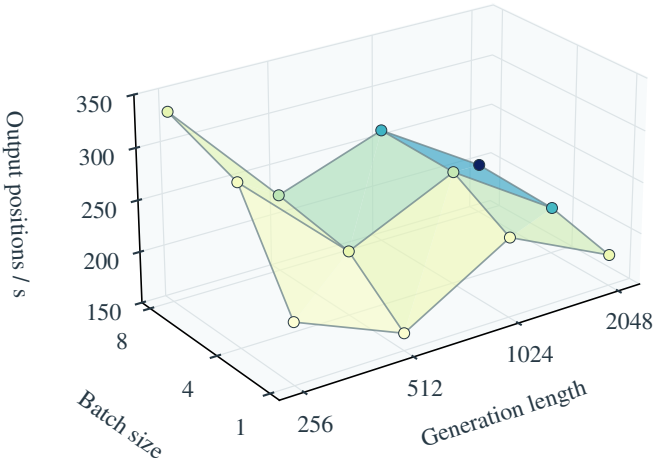
Figure 8: Length–batch scaling across four system metrics and 16 measured conditions on a single H200 with a fixed 66-token prefix replicated across batch. Curves connect measured lengths. Latency and throughput refer to decoding; NFE counts whole-batch model calls. Error bars are pointwise 95% bootstrap intervals over repeated measurements of the same workload. Memory is the maximum allocated peak from five separate memory repeats. The original decoder uses block length 32; shared TACG uses 64.

The profiles show how generated length and batching interact with throughput and memory on this workload.

F Algebraic properties of temporal support

This appendix records algebraic identities for TACG's TILG readout. The derivations describe the implemented score and its role in commitment control; they do not assert posterior recovery or an optimal stopping rule.

(a) Throughput and memory



(b) Measured values

Batch size \ Generation length	256	512	1024	2048
1	209.8 / s 16.2 GiB	166.7 / s 17.1 GiB	227.5 / s 19.1 GiB	180.1 / s 22.9 GiB
4	306.8 / s 19.8 GiB	212.1 / s 23.7 GiB	259.5 / s 31.4 GiB	195.3 / s 46.8 GiB
8	333.0 / s 24.7 GiB	224.8 / s 32.4 GiB	261.1 / s 47.8 GiB	198.5 / s 78.7 GiB

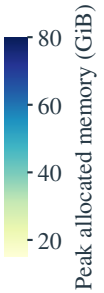


Figure 9: TACG throughput and memory on the measured length–batch grid. (a) Twelve nodes with length on a base-2 logarithmic axis. Translucent triangular faces connect measured vertices without adding observations; face colors encode their mean peak allocated memory. (b) All measured throughput and memory values. Throughput divides total generated output positions by total decode time; node/matrix colors show the largest allocated-memory peak across memory repeats.

F.1 Temporal support is a historical log-ratio signal

For derivation, write the auxiliary readout scores as

$$u_t = (1 + w)z_t - w\bar{z}_{t-1}. \tag{11}$$

Let $p_t = \mathrm{softmax}(z_t)$, $p_t^{\mathrm{ref}} = \mathrm{softmax}(\bar{z}_{t-1})$, and $q_t = \mathrm{softmax}(u_t)$. Since logits are identifiable only up to an additive constant, we can write

$$\exp u_t(v) = \exp\left((1 + w)z_t(v) - w\bar{z}_{t-1}(v)\right) \tag{12}$$

$$\propto \frac{p_t(v)^{1+w}}{p_t^{\mathrm{ref}}(v)^w}. \tag{13}$$

After normalization,

$$q_t(v) \propto p_t(v)\left(\frac{p_t(v)}{p_t^{\mathrm{ref}}(v)}\right)^w. \tag{14}$$

The auxiliary readout therefore exposes a likelihood ratio between the current belief and the historical reference. Pairwise log-odds changes enter this readout explicitly. The bounded probability-gain score in Eq. 6 additionally depends on normalization across the vocabulary.

F.2 First-order belief-innovation interpretation

Let $d_t = z_t - \bar{z}_{t-1}$ and $q_w = \mathrm{softmax}(z_t + w d_t)$. At $w = 0$, $q_w = p_t$. The derivative of softmax gives

$$\left.\frac{\partial q_w(v)}{\partial w}\right|_{w=0} = p_t(v)\left(d_t(v) - \mathbb{E}_{u\sim p_t}[d_t(u)]\right). \tag{15}$$

Thus a small increase in w raises the auxiliary readout value of tokens whose belief innovation is above the current average and lowers the readout value of tokens whose innovation is below that average. This supports the use of q_t as a directional temporal signal and distinguishes it from confidence sharpening.

F.3 Gauge invariance

Categorical logits are not uniquely identified: adding a token-independent constant does not change the softmax. Suppose $z'_t(v) = z_t(v) + a$ and $\bar{z}'_{t-1}(v) = \bar{z}_{t-1}(v) + b$ for constants a and b that do not depend on token v. Then

$$u'_t(v) = (1+w)z'_t(v) - w\bar{z}'_{t-1}(v) = u_t(v) + (1+w)a - wb. \tag{16}$$

Normalization removes the added term because it is identical for every token, so q_t remains unchanged. The temporal support statistic is therefore invariant to the additive gauge of logits at each position.

F.4 Difference from KL-style stability signals

Define the token-wise log-ratio

$$r_t(v) = \log p_t(v) - \log p_t^{\mathrm{ref}}(v). \tag{17}$$

A KL-style stability score aggregates this signal over the whole distribution,

$$D_{\mathrm{KL}}(p_t \| p_t^{\mathrm{ref}}) = \mathbb{E}_{v\sim p_t}[r_t(v)]. \tag{18}$$

This quantity measures distribution-level change. A candidate-level diagnostic of the same relative history is

$$r_t(\hat{x}_i) = \log p_t(i, \hat{x}_i) - \log p_t^{\mathrm{ref}}(i, \hat{x}_i). \tag{19}$$

The implemented probability gain b_i also depends on the full normalization and is not generally a monotone function of this scalar log-ratio. The two forms of information answer different questions. KL-style stability asks whether the whole distribution is still changing. TILG supplies a signed candidate-conditioned contrast, while b_i retains its positive gain for the readiness score. A distribution may change substantially while the current proposal is consistently reinforced, and a distribution may remain nearly unchanged while no candidate becomes more commit-ready. This distinguishes candidate-conditioned temporal support from distribution-level stability.