

TICD: Time-Aware Internal Contrastive Decoding for Enhancing Factuality in Diffusion Language Models

Zike Chen¹, Hongkun Dou^{2,†}, Mukang You¹, Yue Deng^{1,2}, Jiajun Li^{1,†}

¹Beihang University, ²Zhongguancun Academy

[†]Correspondence: douhk@buaa.edu.cn, jiajunli@buaa.edu.cn

Abstract

Diffusion large language models (dLLMs) generate text through iterative denoising, but they remain prone to factual errors in knowledge-intensive tasks. Through layer-wise and timestep-wise probing, we observe a *spatiotemporal stratification* pattern in dLLMs: structural-token predictions tend to converge earlier across layers, whereas factual-token predictions rely more heavily on deep layers and become reliable only after sufficient denoising progress. Motivated by this observation, we propose *Time-Aware Internal Contrastive Decoding* (TICD), a training-free decoding method that calibrates final-layer predictions using entropy-gated intermediate-layer references only during semantically reliable denoising stages. This design uses reliable final-intermediate prediction gaps as correction signals to strengthen factual predictions at the final layer. Experiments on LLaDA-8B and Dream-7B across factuality and reasoning benchmarks show consistent improvements over vanilla decoding.

1 Introduction

Diffusion large language models (dLLMs) (Nie et al., 2025; Ye et al., 2025; Zhu et al., 2025) have emerged as a competitive alternative to autoregressive (AR) models (Lou et al., 2024; Shi et al., 2024; Sahoo et al., 2024). By framing text generation as iterative denoising over masked sequences, dLLMs replace strict left-to-right decoding with parallel updates of arbitrary token positions, which opens new opportunities for global planning, structured generation, and iterative self-correction.

However, factual reliability remains a central challenge for dLLMs. In knowledge-intensive tasks, they can still produce fluent but incorrect answers, inheriting the hallucination problem widely observed in language generation (Ji et al., 2023). Existing mitigation strategies for dLLMs mostly operate outside the model’s internal prediction

structure, including reinforcement learning over denoising trajectories (Zhao et al., 2025; Yang et al., 2025; Gong et al., 2025) and inference-time re-masking or trajectory-search heuristics (Wang et al., 2025a; Huang et al., 2026). These methods can improve outputs, but they often require additional optimization, extra sampling paths, or carefully tuned schedules.

This raises a question: can a dLLM use its own internal prediction dynamics to improve factuality at inference time, without retraining, external rewards, or additional search trajectories?

To answer this question, we systematically analyze how internal predictions evolve across both Transformer depth and denoising timesteps. Our analysis reveals a *spatiotemporal stratification* pattern in dLLMs. Along the depth dimension, a quantitative token-type analysis shows that structural-token predictions tend to converge earlier across layers, whereas factual-token predictions rely more heavily on deep layers. Along the denoising-time dimension, timestep-wise probing shows that final-layer factual confidence becomes reliable only after sufficient denoising progress. These results indicate that factual prediction in dLLMs is jointly shaped by layer depth and denoising progress. This further suggests that layer-wise discrepancies provide useful correction signals only when the denoising state is semantically mature.

Based on this observation, we propose *Time-Aware Internal Contrastive Decoding* (TICD), a training-free inference-time decoding method for dLLMs. TICD calibrates the final-layer prediction using intermediate-layer references, but activates this contrastive correction only in late denoising stages where the prediction gap is reliable. An entropy-gated weighting mechanism further down-weights uncertain intermediate references, preventing high-entropy early-exit distributions from dominating the correction.

We evaluate TICD across factuality and reason-

ing benchmarks, including TruthfulQA, GSM8K, SVAMP, and Countdown. Experiments on LLaDA-8B and Dream-7B demonstrate that TICD consistently improves truthfulness and reasoning accuracy while preserving inference efficiency. We further show that TICD is complementary to other inference-time decoding methods, suggesting that logit-level calibration and trajectory-level sampling improvements can be combined.

Contributions. Our contributions are summarized as follows:

- We identify and quantitatively validate *spatiotemporal stratification* in dLLMs, showing that structural-token and factual-token predictions mature differently across layer depth and denoising progress.
- We propose *TICD*, a training-free internal decoding method that uses time-aware logit calibration to improve factuality and reasoning consistency.
- We provide a mechanistic interpretation of TICD as an internal analogue of Classifier-Free Guidance (CFG), clarifying how intermediate-layer references help amplify factual content in the final-layer prediction.
- We demonstrate consistent gains across factuality and reasoning benchmarks on LLaDA-8B and Dream-7B, and show that TICD can be integrated with other inference-time decoding methods for further improvements.

2 Related Work

Diffusion Language Models.

Diffusion language modeling began with probabilistic formulations that generate discrete text through corruption and denoising processes (Austin et al., 2021), later evolving into continuous-time and score-based objectives such as SEDD (Lou et al., 2024). Recent large-scale dLLMs, including Dream (Ye et al., 2025) and LLaDA (Nie et al., 2025), demonstrate that diffusion-based generation can serve as a competitive text generation paradigm. Existing dLLM work has primarily studied training objectives, sampling efficiency, and re-masking schedules (Wang et al., 2025a). Our work instead

focuses on how internal layer-wise predictions become reliable during denoising and how this temporal structure can be used for factuality-oriented decoding.

Post-Training and Inference-Time Methods.

The rapid scaling of dLLMs has catalyzed research into post-training alignment and advanced inference strategies. One major avenue is reinforcement learning (RL), where variants like GRPO (Shao et al., 2024; DeepSeek-AI, 2025) optimize denoising trajectories using external rewards, though they incur significant training overhead. Alternatively, *test-time inference strategies* (Wei et al., 2022; Madaan et al., 2023; Snell et al., 2024; Yao et al., 2023; Liu et al., 2025) have been widely adopted to enhance the reliability of LLMs without parameter updates. A simple yet effective approach is to leverage the internal architecture of Transformers for decoding. For instance, in autoregressive (AR) settings, *layer-wise contrastive decoding* such as DoLa (Chuang et al., 2024) and SLED (Zhang et al., 2024) mitigates hallucinations by contrasting shallow and deep layer representations. While effective, these methods assume a static spatial hierarchy.

In the context of dLLMs, recent studies reveal that *answers evolve across denoising timesteps* (Lou et al., 2024), where correct signals often emerge mid-trajectory but may be overwritten by later noise (Li et al., 2025; Wang et al., 2025b). This phenomenon suggests that *inter-layer semantic relationships are inherently dynamic throughout the denoising process*. Recent inference-time work on dLLMs has mainly explored interventions at the trajectory or timestep level. Some methods adjust or search denoising trajectories through re-masking or Monte Carlo sampling (Wang et al., 2025a; Huang et al., 2026), while others aggregate intermediate decoded outputs across timesteps through temporal voting (Wang et al., 2025b). Unlike these approaches, TICD keeps the native sampling trajectory unchanged and does not aggregate predictions across timesteps. Instead, it performs logit-level calibration within each denoising step, aligning internal layer-wise guidance with the temporal maturation of factual signals.

3 Preliminaries and Analysis

3.1 Diffusion Large Language Models

Diffusion large language models (dLLMs) formulate text generation as an iterative denoising process

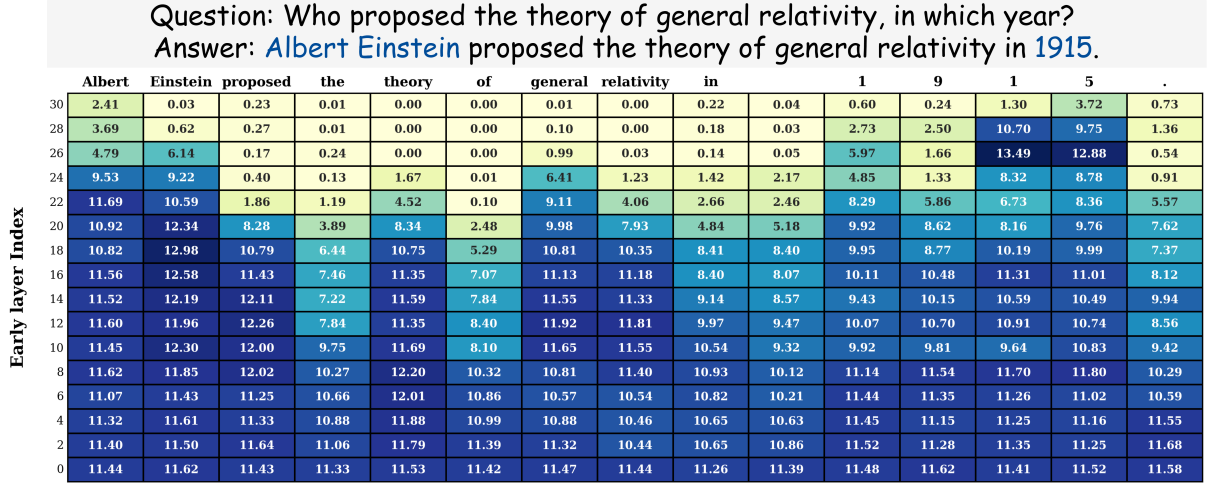


Figure 1: **Spatiotemporal Stratification.** Heatmap of average token-wise KL divergence $\mathcal{D}_{\text{KL}}(P_i^{(N)} \parallel P_i^{(l)})$ across Transformer layers. **Observation:** Factual tokens (e.g., “Albert Einstein”, “1915”) remain highly divergent until the deepest layers, while structural tokens (e.g., “the”, “of”) converge in intermediate layers. This pattern motivates the phase-aware internal contrast in TICD.

over a discrete state space. Let $\mathbf{x}_0 \sim p_{\text{data}}$ denote a clean sequence of length L , and let $\mathbf{x}_t$ denote its noisy version at denoising timestep $t \in [0, T]$.

Forward process. The forward process is a Markov chain $q(\mathbf{x}_{1:T} \mid \mathbf{x}_0) = \prod_{t=1}^T q(\mathbf{x}_t \mid \mathbf{x}_{t-1})$ that progressively replaces tokens of $\mathbf{x}_0$ with a special mask symbol [MASK], until the sequence is fully masked at $t = T$.

Reverse process. The generative process reverses the forward process via a Markov chain $p_{\theta}(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^T p_{\theta}(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$. The exact reverse transition $p_{\theta}(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$ is intractable, so we approximate it with the tractable forward posterior conditioned on a model-predicted clean sequence $\hat{\mathbf{x}}_0$:

$$p_{\theta}(\mathbf{x}_{t-1} \mid \mathbf{x}_t) = \sum_{\hat{\mathbf{x}}_0} q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \hat{\mathbf{x}}_0) p_{\theta}(\hat{\mathbf{x}}_0 \mid \mathbf{x}_t), \quad (1)$$

Here, $p_{\theta}(\hat{\mathbf{x}}_0 \mid \mathbf{x}_t) = \text{Softmax}(f_{\theta}(\mathbf{x}_t, t))$ is the per-position categorical distribution produced by the denoising network $f_{\theta}(\mathbf{x}_t, t)$, a Transformer with N stacked layers; and $q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \hat{\mathbf{x}}_0)$ is the forward posterior, typically implemented by re-masking a subset of $\hat{\mathbf{x}}_0$.

Eq. (1) factorizes each generative step into two operations: a *prediction* step that produces $p_{\theta}(\hat{\mathbf{x}}_0 \mid \mathbf{x}_t)$, and a *re-masking* step that draws $\mathbf{x}_{t-1}$ from the forward posterior. The re-masking step is commonly realized by the low-confidence strategy (Nie et al., 2025), which keeps high-confidence tokens in $\hat{\mathbf{x}}_0$ and re-masks the rest. Crucially, TICD in-

tervenes purely in the prediction step: it calibrates the predicted distribution $p_{\theta}(\hat{\mathbf{x}}_0 \mid \mathbf{x}_t)$ before the re-masking step consumes it.

3.2 Depth-wise Stratification

The single-step denoising network $f_{\theta}(\mathbf{x}_t, t)$ in dLLMs is typically parameterized as a Transformer with N stacked layers. Recent studies indicate that Transformers capture linguistic features hierarchically: lower layers focus on surface-level syntax, while abstract semantic and factual knowledge emerge in deeper layers (Tenney et al., 2019; Chuang et al., 2024). However, the specific dynamics of how factual knowledge matures within the iterative discrete diffusion trajectory remain underexplored.

To probe this interaction, we use an early-exit strategy (Teerapittayanon et al., 2016; Elbayad et al., 2019; Schuster et al., 2022). For each layer l , we apply the shared LM head $\phi(\cdot)$ to the hidden states $\mathbf{H}^{(l)} \in \mathbb{R}^{L \times d}$ to obtain a per-position prediction $P_i^{(l)} = \text{Softmax}(\phi(\mathbf{h}_i^{(l)}))$ at each token position i . We then measure how far each intermediate prediction is from the final-layer prediction $P_i^{(N)}$ via the KL divergence:

$$\mathcal{D}_{\text{KL}}(P_i^{(N)} \parallel P_i^{(l)}) = \sum_{v \in \mathcal{V}} P_i^{(N)}(v \mid \mathbf{x}_t) \log \frac{P_i^{(N)}(v \mid \mathbf{x}_t)}{P_i^{(l)}(v \mid \mathbf{x}_t)}. \quad (2)$$

We run this probing analysis on LLaDA-8B-Instruct (Nie et al., 2025) with a QA task. We define the mask ratio as $\gamma = L_{\text{mask}}/L_{\text{gen}}$, where L_{mask} is the number of masked positions and L_{gen}

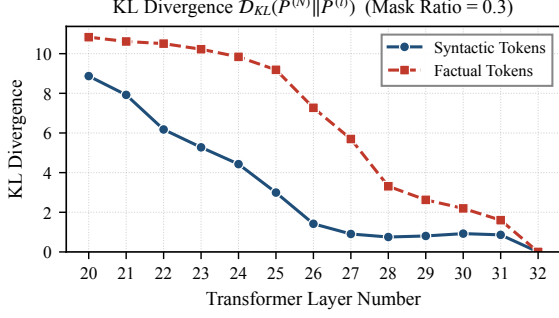


Figure 2: **Layer-wise KL by token type** ($\gamma = 0.3$). Average $\mathcal{D}_{\text{KL}}(P^{(N)} \parallel P^{(l)})$ over all probing samples, separated by token type. Structural tokens (e.g., adpositions, determiners, conjunctions) flatten around layer 26, while factual tokens (entities, numbers, content nouns) keep diverging from $P^{(N)}$ until the last few layers.

is the generated sequence length. We fix $\gamma = 0.3$ so that the model operates in the late-denoising stage, where most context tokens have been resolved. Given a question and its reference answer, we mask only the answer tokens and inspect how the model recovers them across layers. Each value in Figure 1 is averaged over 50 Monte Carlo runs.

Figure 1 visualizes the resulting per-token KL across layers. For structural tokens such as “the” and “of”, $P_i^{(l)}$ aligns with $P_i^{(N)}$ already in mid layers. For factual-token positions such as “Albert Einstein”, the KL stays high through most of the network and drops only in the last few layers. These results show that, in the late-denoising stage, structural information is already established in shallower layers, while factual content is primarily resolved in deep layers.

The heatmap visualizes a few representative examples. To verify that the same pattern holds at the corpus level, we run a token-type analysis on 500 TruthfulQA trajectories under the same late-stage mask ratio ($\gamma = 0.3$). We group tokens into *structural tokens* (e.g., adpositions, determiners, conjunctions) and *factual tokens* (e.g., proper nouns, named entities, numbers). For each group, we average the per-token KL $\mathcal{D}_{\text{KL}}(P_i^{(N)} \parallel P_i^{(l)})$ over all positions and trajectories, and plot the resulting layer-wise curves in Figure 2. The structural curve plateaus in mid layers, while the factual curve remains highly divergent and only sharpens in the terminal layers. The corpus-level statistics therefore confirm the per-example pattern shown in the heatmap.

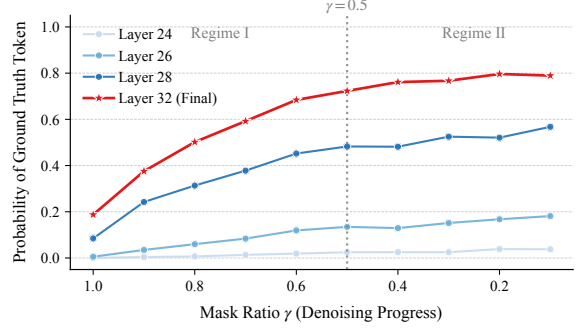


Figure 3: **Evolution of layer-wise probability assigned to the ground-truth token on TruthfulQA, as the mask ratio γ decreases.** Each point is averaged over 500 samples. In the early-denoising stage (large γ , Regime I), all layers assign low probability to the ground-truth token. In the late-denoising stage (small γ , Regime II), the final layer (Layer 32) assigns substantially higher probability than intermediate layers (Layers 24–28).

3.3 Denoising-Time Stratification

Unlike autoregressive models, a dLLM predicts under a context whose completeness changes over time. When γ is large, most positions are still masked and the visible context is sparse, so layer-wise differences mostly reflect predictive uncertainty. When γ is small, the visible context is rich enough for deep layers to commit to factual predictions, and layer-wise differences begin to encode actual factual refinement. The usefulness of layer-wise discrepancies in dLLMs therefore depends on both depth and denoising progress.

Protocol. To investigate how layer-wise confidence evolves over denoising progress, we conduct a probing analysis on TruthfulQA (Lin et al., 2022) using LLaDA-8B-Instruct. For each sample, we concatenate the question with its best correct answer and mask only the answer tokens at different mask ratios $\gamma \in [0.1, 1.0]$. We then feed the masked sequence into the model and record the probability assigned to the ground-truth answer tokens at different layers. This allows us to track how the confidence gap between intermediate layers and the final layer changes as denoising progresses.

Figure 3 shows how the probability assigned to the ground-truth token changes with the mask ratio γ across layers. The evolution reveals two stages of denoising.

In the early-denoising stage (large γ ; Regime I), all layers assign low probability to the ground-truth token. This indicates that the model has not yet

accumulated enough visible context for reliable factual prediction. At this stage, discrepancies between intermediate and final layers mainly reflect predictive uncertainty under partial context, rather than meaningful factual differences. Applying contrastive correction here would risk amplifying unstable signals.

In the late-denoising stage (small γ ; Regime II), the final layer assigns substantially higher probability to the ground-truth token than intermediate layers. This suggests that deeper layers have incorporated richer contextual evidence and can distinguish factual candidates more reliably. The final-intermediate gap in this stage therefore reflects factual refinement contributed by deep layers, making it a suitable correction signal.

This temporal pattern explains why static layer-wise correction is not appropriate for dLLMs. The same layer-wise discrepancy can be uncertainty-dominated in Regime I but reflect factual refinement in Regime II. TlCD therefore activates internal contrast only after the denoising state enters the late-denoising stage, where the final-intermediate gap becomes reliable for factual calibration.

4 Method

Building on the spatiotemporal stratification analysis in Section 3, we propose **Time-Aware Internal Contrastive Decoding (TlCD)**, a training-free inference-time decoding method for dLLMs. As illustrated in Figure 4, TlCD is built around two coordinated components: (1) *internal contrast*, which calibrates the final-layer prediction using intermediate-layer references, and (2) a *phase-aware gate*, which activates this calibration only after γ falls below a threshold.

4.1 Entropy-Gated Internal Contrast

The depth-wise analysis in Section 3.2 shows that intermediate layers already capture structural preferences, while factual predictions are refined mainly in deeper layers. This motivates using intermediate-layer predictions as internal references. By contrasting the final-layer prediction with these references, TlCD downweights preferences already shared by intermediate and final layers and amplifies factual refinements that emerge in deeper layers.

Let $\mathbf{z}_i^{(l)} = \log P_i^{(l)}$ denote the log-probability vector at token position i from layer l . Let $\mathcal{L}_{\text{cand}} \subset \{1, \dots, N-1\}$ be a set of candidate intermedi-

ate layers. TlCD calibrates the final-layer log-probabilities at position i as

$$\tilde{\mathbf{z}}_i^{(N)} = \mathbf{z}_i^{(N)} + \lambda(t) \sum_{l \in \mathcal{L}_{\text{cand}}} w_{l,i} (\mathbf{z}_i^{(N)} - \mathbf{z}_i^{(l)}), \quad (3)$$

where $\lambda(t)$ is the phase-aware gating coefficient (Section 4.2) and $w_{l,i}$ is the weight assigned to layer l at position i . In the contrastive difference $\mathbf{z}_i^{(N)} - \mathbf{z}_i^{(l)}$, structural preferences shared by the intermediate and final layers are largely subtracted out. Factual content that is weak in the intermediate-layer reference but strong in the final-layer prediction remains in the difference. Adding this weighted difference back to the final-layer prediction therefore downweights shared structural preferences and amplifies final-layer factual content.

Not every intermediate layer is a reliable reference. A useful reference should both differ meaningfully from the final-layer prediction and be confident enough to serve as a stable reference. TlCD therefore assigns $w_{l,i}$ using both layer-wise divergence and prediction entropy: KL divergence measures how much the reference differs from the final layer, while entropy penalizes uncertain intermediate predictions.

$$w_{l,i} = \frac{\exp(\mathcal{D}_{\text{KL}}(P_i^{(N)} \| P_i^{(l)}) \cdot (1 - \bar{\mathcal{H}}(P_i^{(l)})) / \tau)}{\sum_{k \in \mathcal{L}_{\text{cand}}} \exp(\mathcal{D}_{\text{KL}}(P_i^{(N)} \| P_i^{(k)}) \cdot (1 - \bar{\mathcal{H}}(P_i^{(k)})) / \tau)}, \quad (4)$$

where $\bar{\mathcal{H}}(P_i^{(l)})$ denotes the normalized entropy of the distribution at layer l and token position i . Specifically, for a vocabulary $\mathcal{V}$, it is defined as:

$$\bar{\mathcal{H}}(P_i^{(l)}) = \frac{-\sum_{v \in \mathcal{V}} P_i^{(l)}(v) \log P_i^{(l)}(v)}{\log |\mathcal{V}|}. \quad (5)$$

By combining KL divergence with the entropy penalty, TlCD favors intermediate-layer references that provide a meaningful contrast while remaining sufficiently confident.

4.2 Phase-Aware Gating and Sparse Computation

Section 3.3 shows that internal contrast provides a reliable correction signal only in the late-denoising stage. We therefore define a phase-aware schedule for $\lambda(t)$:

$$\lambda(t) = \alpha \cdot \mathbb{I}(\gamma \leq \gamma_{\text{semantic}}), \quad (6)$$

where α is the calibration strength and γ_{semantic} is the threshold separating the two stages. Equivalently, calibration is inactive in Regime I and active with strength α in Regime II.

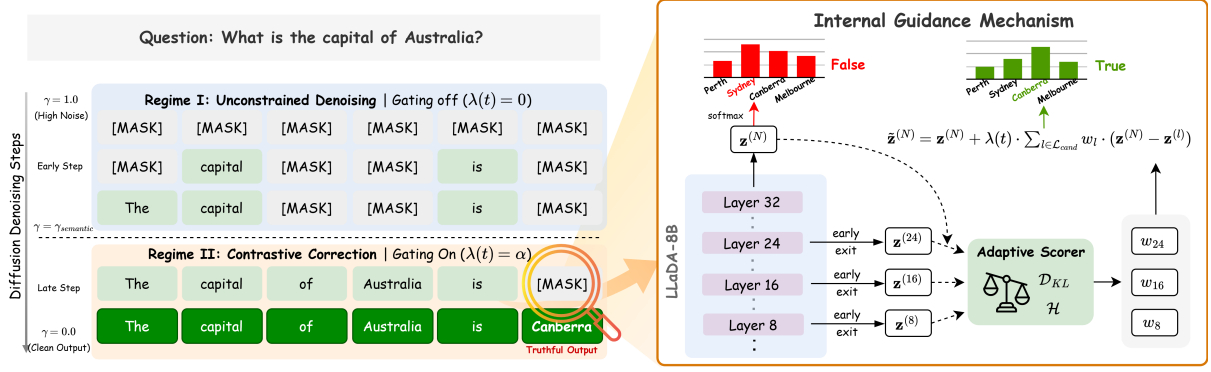


Figure 4: **Overview of TlCD.** **Left (temporal dynamics):** the denoising process is split by the mask ratio γ into Regime I (early-denoising stage; calibration inactive) and Regime II (late-denoising stage; calibration active). **Right (internal contrast):** for masked positions, intermediate-layer predictions $\mathbf{z}_i^{(l)}$ are obtained via early exit; an adaptive scorer weights each layer by KL divergence and prediction entropy to produce the calibrated final prediction $\tilde{\mathbf{z}}^{(N)}$.

The phase-aware schedule also keeps the computational overhead small. In Regime I, calibration is inactive and the model follows the vanilla denoising path: no intermediate-layer projection, no layer scoring, and no adaptive layer weighting. In Regime II, the LM head is applied to intermediate hidden states only at masked positions $i \in \mathcal{M}$ (i.e., positions still to be resolved at step t), so the projection cost scales with $|\mathcal{M}|$ rather than the full sequence length. The two sources of sparsity (temporal: no work in Regime I; spatial: only masked positions in Regime II) keep TlCD’s added latency under 15%, as measured in Section 5.4.

The calibration in Eq. (3) can occasionally increase the scores of tokens assigned very low probability by the final layer. To prevent this, we apply the Adaptive Plausibility Constraint (APC) of Li et al. (2023): at each masked position i , we restrict $\tilde{\mathbf{z}}_i^{(N)}$ to the top- k candidates of $P_i^{(N)}$ (with $k = \eta \cdot |\mathcal{V}|$) and set the remaining entries to $-\infty$. APC keeps the calibrated distribution inside a locally plausible region of the base model. Algorithm 1 in Appendix B gives the full procedure.

4.3 Mechanistic Interpretation: Connection to Internal Guidance

TlCD can be interpreted as a single-pass internal analogue of Classifier-Free Guidance (CFG) (Ho and Salimans, 2022) and Autoguidance (Karras et al., 2024). Standard CFG contrasts a conditional prediction with an external unconditional model, while Autoguidance shows that a weaker version of the same model can also provide a useful reference. TlCD follows this intuition internally: intermediate-layer predictions serve as internal ref-

erences, while the final-layer prediction provides the target to be calibrated. Let the weighted aggregate of intermediate-layer references at position i be $\mathbf{z}_i^{(\text{ref})} = \sum_l w_{l,i} \mathbf{z}_i^{(l)}$. Under $\sum_l w_{l,i} = 1$, the update rule in Eq. (3) can be reformulated as:

$$\tilde{\mathbf{z}}_i^{(N)} = \mathbf{z}_i^{(\text{ref})} + (1 + \lambda(t)) \cdot (\mathbf{z}_i^{(N)} - \mathbf{z}_i^{(\text{ref})}). \quad (7)$$

This form shows that TlCD amplifies the gap between the final-layer prediction and the intermediate-layer reference. Consistent with the probing results in Section 3, this gap emphasizes deeper-layer factual refinements, while shared structural preferences receive little additional gain.

5 Experiments

Datasets & Metrics. We evaluate TlCD on two task categories. For **factualty**, we use TruthfulQA (Lin et al., 2022) and report MC1, MC2, and MC3 (Chuang et al., 2024). For **reasoning**, we use GSM8K (Cobbe et al., 2021), SVAMP (Patel et al., 2021), and Countdown (Ye et al., 2024), and report accuracy on all three benchmarks. Detailed dataset statistics are provided in Appendices A.1 and A.4.

Models & Baselines. We evaluate TlCD on two dLLM models, LLaDA-8B-Instruct (Nie et al., 2025) and Dream-7B-Instruct (Ye et al., 2025), and compare it with the following inference-time baselines: (i) Vanilla Sampling; (ii) DoLa (Chuang et al., 2024), adapted to the diffusion process via step-wise JSD selection; (iii) SLED (Zhang et al., 2024), re-implemented for diffusion trajectories; (iv) ReMDM (Wang et al., 2025a), a trajectory-level inference method based on re-masking; and (v) Temporal Self-Consistency Vot-

ing (TSCV) (Wang et al., 2025b), a temporal voting method over intermediate decoded outputs.

TICD Configuration. For multiple-choice tasks, we set $\alpha = 2.5$ and $\gamma_{\text{semantic}} = 0.5$. For reasoning tasks, we tune α and γ_{semantic} in $[0, 1.0]$. The softmax temperature τ is fixed at 0.5 across all tasks. Candidate-layer selection and detailed configurations are provided in Appendix A.3.

5.1 Evaluation on Factuality and Reasoning Benchmarks

We present the main quantitative comparison of TICD against the vanilla baseline and logit-level inference-time interventions in Table 1.

Factuality (TruthfulQA). TICD improves truthfulness on both models. On LLaDA-8B, TICD gives the largest gain on MC1, improving over the baseline by 5.7%, and also improves MC2 by 5.5%. On Dream-7B, TICD again improves MC1 and MC2. While DoLa remains competitive on MC3, TICD shows a clear advantage on MC1 and MC2. This suggests that TICD is more effective at selecting the most truthful answer than at merely improving broad ranking over answer candidates. We also evaluate TICD in an open-ended QA setting using TriviaQA, where it outperforms vanilla decoding; detailed experimental settings and results are reported in Appendix C.

Reasoning (GSM8K, SVAMP, Countdown). Inference-time interventions should improve generation quality without disrupting reasoning. Table 1 shows that static layer-wise baselines such as DoLa and SLED can hurt reasoning in the diffusion setting. For example, DoLa drops Dream-7B GSM8K from 75.1% to 55.3%, suggesting that layer-wise subtraction without temporal control can interfere with multi-step reasoning. TICD avoids this failure mode through phase-aware gating, and outperforms the vanilla baseline across all reasoning benchmarks on both models.

5.2 Ablation Study

Effect of phase-aware gating. We first examine the effectiveness of phase-aware gating. To do so, we vary the mask threshold γ_{semantic} from 0.1 to 1.0 on SVAMP using LLaDA-8B. As shown in Figure 5, accuracy follows a non-monotonic trend. When γ_{semantic} is too small, TICD is activated only at the very end of denoising and has limited room to correct errors. As the threshold increases, accuracy improves and peaks at $\gamma_{\text{semantic}} = 0.7$. When $\gamma_{\text{semantic}} = 1.0$, TICD is applied throughout the

full denoising trajectory, and accuracy drops below the vanilla baseline. This supports our temporal analysis: applying contrastive calibration too early amplifies uncertainty-dominated layer discrepancies, while late-stage activation provides a more reliable factual correction signal.

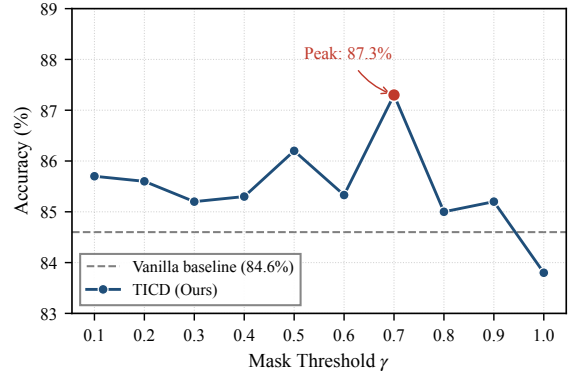


Figure 5: Effect of the mask threshold γ_{semantic} on SVAMP accuracy. Performance peaks at $\gamma_{\text{semantic}} = 0.7$, while full-range intervention ($\gamma_{\text{semantic}} = 1.0$) degrades below the vanilla baseline.

Effect of entropy-gated weighting. We next evaluate the layer-weighting mechanism on GSM8K and SVAMP using LLaDA-8B. We compare full TICD with two alternatives: *Uniform Averaging*, which assigns equal weights to candidate intermediate layers, and *KL-only Selection*, which removes the entropy term ($1 - \bar{H}$) from Eq. (4). As shown in Table 2, full TICD performs best on both benchmarks. Uniform averaging brings only small gains, and KL-only weighting does not consistently improve over the vanilla baseline. This confirms the role of the entropy term: KL divergence selects references that differ from the final layer, while entropy filters out uncertain intermediate-layer predictions.

5.3 Evaluation on Integrating TICD with Other Inference-Time Decoding Methods

TICD calibrates prediction logits within each denoising step, without changing the sampling trajectory or temporal aggregation procedure. It can therefore be integrated with other inference-time decoding methods. To verify this, we combine TICD with ReMDM and TSCV, and report the LLaDA-8B results in Table 3.

Table 3 shows that TICD can be integrated with other decoding methods for additional gains. When combined with ReMDM, TICD achieves the best results on both GSM8K and SVAMP. TSCV is

Table 1: Main results on factuality and reasoning. We report MC1/MC2/MC3 on TruthfulQA and accuracy (%) on GSM8K, SVAMP, and Countdown. Blue background and bold indicate the best result within each model group; green numbers show absolute improvement of TICD over the baseline.

Model	Method	TruthfulQA			Reasoning		
		MC1 $\uparrow$	MC2 $\uparrow$	MC3 $\uparrow$	GSM8K $\uparrow$	SVAMP $\uparrow$	Countdown $\uparrow$
LLaDA-8B	Baseline	31.6 $\pm$ 0.2	49.4 $\pm$ 0.1	23.9 $\pm$ 0.4	78.0 $\pm$ 0.2	84.6 $\pm$ 0.3	22.3 $\pm$ 0.3
	+DoLa	36.4 $\pm$ 0.1	52.5 $\pm$ 0.3	27.7$\pm$0.2	74.0 $\pm$ 0.3	78.8 $\pm$ 0.0	12.5 $\pm$ 0.4
	+SLED	35.0 $\pm$ 0.3	54.2 $\pm$ 0.2	27.0 $\pm$ 0.1	78.2 $\pm$ 0.2	84.7 $\pm$ 0.3	24.2 $\pm$ 0.2
	+TICD	37.3$\pm$0.1	54.9$\pm$0.2	27.3 $\pm$ 0.3	80.1$\pm$0.0	87.3$\pm$0.2	26.2$\pm$0.3
		+5.7	+5.5	+3.4	+2.1	+2.7	+3.9
Dream-7B	Baseline	34.6 $\pm$ 0.2	53.4 $\pm$ 0.1	24.5 $\pm$ 0.3	75.1 $\pm$ 0.3	81.4 $\pm$ 0.2	14.8 $\pm$ 0.4
	+DoLa	35.1 $\pm$ 0.1	56.0 $\pm$ 0.3	29.4$\pm$0.2	55.3 $\pm$ 0.4	61.4 $\pm$ 0.3	10.5 $\pm$ 0.3
	+SLED	34.9 $\pm$ 0.2	52.9 $\pm$ 0.0	26.7 $\pm$ 0.2	73.8 $\pm$ 0.3	76.6 $\pm$ 0.3	22.0$\pm$0.1
	+TICD	36.4$\pm$0.2	57.2$\pm$0.3	28.1 $\pm$ 0.2	76.0$\pm$0.1	83.7$\pm$0.3	20.3 $\pm$ 0.2
		+1.8	+3.8	+3.6	+0.9	+2.3	+5.5

Table 2: Ablation of the layer-weighting mechanism on LLaDA-8B.

Mechanism	GSM8K $\uparrow$	SVAMP $\uparrow$
Vanilla Baseline	78.0 $\pm$ 0.2	84.6 $\pm$ 0.3
Uniform Averaging	78.5 $\pm$ 0.3	85.0 $\pm$ 0.2
KL-only Selection	78.9 $\pm$ 0.2	84.7 $\pm$ 0.3
TICD (Full)	80.1$\pm$0.0	87.3$\pm$0.2

Table 3: Results of integrating TICD with ReMDM and TSCV on LLaDA-8B.

Method	GSM8K $\uparrow$	SVAMP $\uparrow$
Baseline	78.0 $\pm$ 0.2	84.6 $\pm$ 0.3
ReMDM	79.9 $\pm$ 0.2	85.5 $\pm$ 0.3
TSCV	78.1 $\pm$ 0.1	84.5 $\pm$ 0.2
TICD	80.1$\pm$0.0	87.3$\pm$0.2
TICD + ReMDM	81.8$\pm$0.2	88.6$\pm$0.3
TICD + TSCV	80.2$\pm$0.1	86.7$\pm$0.3

weaker as a standalone method in this setting, but TICD + TSCV still improves clearly over TSCV alone. These results suggest that TICD provides a logit-level calibration signal that can complement decoding methods operating outside the logit-calibration step.

5.4 Inference Efficiency

We compare the inference cost of TICD with the logit-level baselines used in Table 1. We measure end-to-end latency on SVAMP using a single NVIDIA A100 GPU, with the mask threshold fixed at $\gamma_{\text{semantic}} = 0.7$.

As shown in Table 4, TICD incurs a modest latency increase on both models (14.4% on LLaDA-8B and 14.5% on Dream-7B), lower than SLED

Table 4: Inference efficiency on SVAMP using a single NVIDIA A100 GPU ($\gamma_{\text{semantic}} = 0.7$).

Model	Method	Latency	Overhead $\downarrow$
LLaDA-8B	Baseline	2.98	—
	SLED	3.70	24.2%
	DoLa	3.91	31.2%
	TICD	3.41	14.4%
Dream-7B	Baseline	2.76	—
	SLED	3.48	26.1%
	DoLa	3.62	31.2%
	TICD	3.16	14.5%

and DoLa. This comes from the same design used for calibration: TICD is active only in the late-denoising stage and applies the LM head only to currently masked positions.

6 Conclusion

We introduced **Time-Aware Internal Contrastive Decoding (TICD)**, a training-free decoding method for improving factuality in diffusion large language models. TICD is motivated by a spatiotemporal stratification pattern: structural-token predictions stabilize in intermediate layers, while factual-token predictions rely more on deep layers and become reliable only after sufficient denoising progress. Based on this observation, TICD calibrates final-layer predictions with entropy-gated intermediate-layer references only when the denoising state provides a reliable correction signal.

Experiments on LLaDA-8B and Dream-7B show consistent gains on factuality and reasoning benchmarks while keeping inference overhead manageable. TICD is also compatible with other inference-

time decoding methods because it operates within each denoising step without changing the sampling trajectory. These results highlight internal prediction dynamics as a useful source of self-correction for dLLMs.

Limitations

TICD is an inference-time decoding method and does not update model parameters. Its effectiveness therefore depends on factual signals already present in the final-layer prediction; it cannot recover facts that are absent from the base model. In addition, TICD calibrates token distributions during decoding but does not explicitly verify generated content against external evidence. Future work could combine TICD with retrieval or verification modules for settings that require stronger factual guarantees.

References

- Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. 2021. Structured denoising diffusion models in discrete state-spaces. *Advances in neural information processing systems*, 34:17981–17993.
- Yung-Sung Chuang, Yujia Xie, Hongyin Luo, Yoon Kim, James R. Glass, and Pengcheng He. 2024. [Dola: Decoding by contrasting layers improves factuality in large language models](#). In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.
- 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. 2021. [Training verifiers to solve math word problems](#). *CoRR*, abs/2110.14168.
- DeepSeek-AI. 2025. [DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning](#). *CoRR*, abs/2501.12948.
- Maha Elbayad, Jiatao Gu, Edouard Grave, and Michael Auli. 2019. Depth-adaptive transformer. *arXiv preprint arXiv:1910.10073*.
- Shansan Gong, Ruixiang Zhang, Huangjie Zheng, Jiatao Gu, Navdeep Jaitly, Lingpeng Kong, and Yizhe Zhang. 2025. Diffucoder: Understanding and improving masked diffusion models for code generation. *arXiv preprint arXiv:2506.20639*.
- Jonathan Ho and Tim Salimans. 2022. [Classifier-free diffusion guidance](#). *CoRR*, abs/2207.12598.
- Zheng Huang, Kiran Ramnath, Yueyan Chen, Aosong Feng, Sangmin Woo, Balasubramaniam Srinivasan, Zhichao Xu, Kang Zhou, Shuai Wang, Haibo Ding, and Lin Lee Cheong. 2026. [Diffusion language model inference with monte carlo tree search](#). In *Findings of the Association for Computational Linguistics: EACL 2026, Rabat, Morocco, March 24-29, 2026*, Findings of ACL, pages 3493–3512. Association for Computational Linguistics.
- Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Yejin Bang, Andrea Madotto, and Pascale Fung. 2023. [Survey of hallucination in natural language generation](#). *ACM Comput. Surv.*, 55(12):248:1–248:38.
- Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611.
- Tero Karras, Miika Aittala, Tuomas Kynkäänniemi, Jaakko Lehtinen, Timo Aila, and Samuli Laine. 2024. [Guiding a diffusion model with a bad version of itself](#). In *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.
- Pengxiang Li, Yefan Zhou, Dilxat Muhtar, Lu Yin, Shilin Yan, Li Shen, Yi Liang, Soroush Vosoughi, and Shiwei Liu. 2025. Diffusion language models know the answer before decoding. *arXiv preprint arXiv:2508.19982*.
- Xiang Lisa Li, Ari Holtzman, Daniel Fried, Percy Liang, Jason Eisner, Tatsunori B Hashimoto, Luke Zettlemoyer, and Mike Lewis. 2023. Contrastive decoding: Open-ended text generation as optimization. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: Long papers)*, pages 12286–12312.
- Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. Truthfulqa: Measuring how models mimic human falsehoods. In *Proceedings of the 60th annual meeting of the association for computational linguistics (volume 1: long papers)*, pages 3214–3252.
- Yexiang Liu, Zekun Li, Zhi Fang, Nan Xu, Ran He, and Tieniu Tan. 2025. Rethinking the role of prompting strategies in llm test-time scaling: A perspective of probability theory. *arXiv preprint arXiv:2505.10981*.
- Aaron Lou, Chenlin Meng, and Stefano Ermon. 2024. [Discrete diffusion modeling by estimating the ratios of the data distribution](#). In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.
- Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegrefe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. [Self-refine: Iterative refinement with self-feedback](#). In *Advances in*

- Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023.*
- Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. 2025. [Large language diffusion models](#). *CoRR*, abs/2502.09992.
- Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are nlp models really able to solve simple math word problems? *arXiv preprint arXiv:2103.07191*.
- Subham Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin Chiu, Alexander Rush, and Volodymyr Kuleshov. 2024. Simple and effective masked diffusion language models. *Advances in Neural Information Processing Systems*, 37:130136–130184.
- Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Tran, Yi Tay, and Donald Metzler. 2022. Confident adaptive language modeling. *Advances in Neural Information Processing Systems*, 35:17456–17472.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. [Deepseekmath: Pushing the limits of mathematical reasoning in open language models](#). *CoRR*, abs/2402.03300.
- Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis Titsias. 2024. Simplified and generalized masked diffusion for discrete data. *Advances in neural information processing systems*, 37:103131–103167.
- Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*.
- Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. 2016. Branchynet: Fast inference via early exiting from deep neural networks. In *2016 23rd international conference on pattern recognition (ICPR)*, pages 2464–2469. IEEE.
- Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. Bert rediscovers the classical nlp pipeline. *arXiv preprint arXiv:1905.05950*.
- Guanghan Wang, Yair Schiff, Subham S. Sahoo, and Volodymyr Kuleshov. 2025a. [Remasking discrete diffusion models with inference-time scaling](#). In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*.
- Wen Wang, Bozhen Fang, Chenchen Jing, Yongliang Shen, Yangyi Shen, Qiuyu Wang, Hao Ouyang, Hao Chen, and Chunhua Shen. 2025b. Time is a feature: Exploiting temporal dynamics in diffusion language models. *arXiv preprint arXiv:2508.09138*.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. [Chain-of-thought prompting elicits reasoning in large language models](#). In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.
- Ling Yang, Ye Tian, Bowen Li, Xincheng Zhang, Ke Shen, Yunhai Tong, and Mengdi Wang. 2025. [Mmada: Multimodal large diffusion language models](#). *CoRR*, abs/2505.15809.
- Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822.
- Jiacheng Ye, Jiahui Gao, Shansan Gong, Lin Zheng, Xin Jiang, Zhenguo Li, and Lingpeng Kong. 2024. Beyond autoregression: Discrete diffusion for complex reasoning and planning. *arXiv preprint arXiv:2410.14157*.
- Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. 2025. [Dream 7b: Diffusion large language models](#). *CoRR*, abs/2508.15487.
- Jianyi Zhang, Da-Cheng Juan, Cyrus Rashtchian, Chun-Sung Ferng, Heinrich Jiang, and Yiran Chen. 2024. [SLED: self logits evolution decoding for improving factuality in large language models](#). In *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.
- Siyan Zhao, Devaansh Gupta, Qinqing Zheng, and Aditya Grover. 2025. d1: Scaling reasoning in diffusion large language models via reinforcement learning. *arXiv preprint arXiv:2504.12216*.
- Fengqi Zhu, Rongzhen Wang, Shen Nie, Xiaolu Zhang, Chunwei Wu, Jun Hu, Jun Zhou, Jianfei Chen, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. 2025. [Llada 1.5: Variance-reduced preference optimization for large language diffusion models](#). *CoRR*, abs/2505.19223.

A Experimental Details

A.1 Datasets and Benchmarks

We conduct comprehensive evaluations across two distinct task categories.

Factuality (TruthfulQA). We utilize the TruthfulQA (Lin et al., 2022) benchmark. Following the implementation in Chuang et al. (2024), we compute the metrics based on the model’s log-likelihood scores:

- **MC1:** Measures whether the best correct answer achieves a higher score than the best incorrect answer.
- **MC2:** The total normalized probability assigned to the set of all correct answers.
- **MC3:** The proportion of correct answers that achieve a higher score than the best incorrect answer.

Reasoning (GSM8K, SVAMP, Countdown).

- **GSM8K** (Cobbe et al., 2021): Contains 8.5K high-quality grade-school math problems requiring multi-step reasoning. We report exact-match accuracy on the test split (1,319 samples).
- **SVAMP** (Patel et al., 2021): A challenge set consisting of 1,000 math word problems with varying linguistic structures, designed to test the robustness of reasoning capabilities.
- **Countdown** (Ye et al., 2024): Requires the model to reach a target number using a given set of input numbers and basic arithmetic operations (+, −, ×, /). This task evaluates the model’s ability to perform constraint satisfaction and look-ahead planning.

A.2 Baseline Implementation Details

Since baselines like DoLa (Chuang et al., 2024) and SLED (Zhang et al., 2024) were originally designed for autoregressive (AR) LLMs, we adapt them for the diffusion generation framework. Specifically, these interventions are applied to the *prediction stage* $p_\theta(\hat{x}_0|x_t)$ at each denoising timestep t , strictly targeting only the currently masked tokens $\mathcal{M}$.

DoLa for Diffusion. DoLa mitigates hallucinations by contrasting the final layer with a dynamic intermediate layer. In our diffusion adaptation:

1. **Layer Selection:** At each timestep t , we compute the Jensen-Shannon Divergence (JSD) between the output distribution of the final layer $P^{(N)}$ and each candidate early layer $P^{(l)}$ on the masked tokens.
2. **Contrastive Subtraction:** We select the layer l^* that maximizes the JSD. The contrastive log-probabilities are obtained by subtracting the log-probabilities of this layer from the final layer: $\log P_{\text{contrast}} = \log P^{(N)} - \log P^{(l^*)}$.
3. **Adaptive Plausibility Constraint (APC):** To ensure linguistic fluency, we apply an APC filter to mask out tokens with low probability in the original final-layer distribution before the subtraction.

SLED for Diffusion. SLED improves factuality by evolving logits towards a latent knowledge estimate via gradient approximation. We implement the “Self Logits Evolution” procedure specifically for diffusion:

1. **Top- k Identification:** For the masked tokens, we first identify the top- k candidate tokens from the final-layer logits $z^{(N)}$. **Tokens outside this top- k set are suppressed (masked with $-\infty$) to ensure stability.**
2. **Gradient Approximation:** We iterate through candidate early layers. For each layer l , we calculate the divergence vector $\Delta z = z^{(l)} - z^{(N)}$. We then compute a weight for this layer based on the **cosine similarity** between Δz and the approximate gradient direction of the target probability ($\nabla_z \log p$). This aligns the evolution direction with the model’s internal latent knowledge.
3. **Logit Update:** The final-layer logits $z^{(N)}$ are updated in-place by subtracting the weighted aggregated gradient step. This update is strictly applied only to the top- k indices.

All methods are tuned on the same validation split with comparable search budgets. DoLa’s APC threshold and SLED’s top- k use the same candidate settings as TIGD. For SLED, we tune the step size of the logit update, referred to as the evolution rate, over $[0, 2]$ and select the configuration with the best validation performance.

A.3 Hyperparameter Configuration

Layer Selection Strategy. We determine the optimal candidate layers $\mathcal{L}_{\text{cand}}$ through empirical verification on a held-out validation set. We observe that the effective layer range for contrastive intervention varies significantly by task nature. Based on the validation performance, we partition the model layers into two distinct regions:

- **Factuality (TruthfulQA):** The validation results indicate that factual knowledge is predominantly located in deeper layers. Consequently, we select candidate layers from the **Upper Half** (e.g., range $[N/2, N]$). Specifically, for **LLaDA-8B** (32 layers), we use $\mathcal{L}_{\text{cand}} = \{16, 18, \dots, 30\}$; for **Dream-7B** (28 layers), we use $\mathcal{L}_{\text{cand}} = \{14, 16, \dots, 26\}$.
- **Reasoning (GSM8K, SVAMP, Countdown):** For reasoning tasks, the validation set shows that intervention in deeper layers can disrupt logical coherence. Therefore, we select layers from the **Lower Half** (e.g., range $[0, N/2]$) to leverage early structural preferences. Specifically, for **LLaDA-8B**, we use $\mathcal{L}_{\text{cand}} = \{4, 6, \dots, 16\}$; for **Dream-7B**, we use $\mathcal{L}_{\text{cand}} = \{4, 6, \dots, 14\}$.

We employ a strided selection (step=2) within these validated ranges to optimize inference efficiency.

Calibration Parameters and APC. Table 5 details the specific hyperparameter configurations for both LLaDA-8B and Dream-7B models across different benchmarks. These parameters were determined empirically based on performance on the validation sets.

To ensure the reliability of the contrastive intervention, we apply an **Adaptive Plausibility Constraint (APC)**:

- For **Reasoning Tasks** (GSM8K, SVAMP, Countdown), we employ an absolute filtering strategy with $k = 20$, restricting the candidate pool to the top-20 tokens to prevent tail-risk amplification.
- For **Multiple-Choice Tasks** (TruthfulQA), we use a relative filtering strategy, retaining the top 1% of the vocabulary to allow for diverse semantic exploration.

Layer-Aggregated Weighting in Practice.

Eq. (4) defines a position-wise layer weight $w_{l,i}$ for each masked position i . In our implementation,

we use a layer-aggregated variant for improved robustness: all currently masked positions in $\mathcal{M}$ share the same layer weights w_l . Specifically, we first average the layer score over masked positions and then apply the softmax over candidate layers:

$$w_l = \frac{\exp(\bar{s}_l/\tau)}{\sum_{k \in \mathcal{L}_{\text{cand}}} \exp(\bar{s}_k/\tau)}, \quad (8)$$

where

$$\bar{s}_l = \frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \mathcal{D}_{\text{KL}}(P_i^{(N)} \| P_i^{(l)}) \cdot (1 - \mathcal{H}(P_i^{(l)})). \quad (9)$$

This aggregation smooths noisy per-position estimates while preserving the same KL-entropy weighting principle as Eq. (4).

Table 5: Hyperparameter settings for TICA across different benchmarks. We report specific α (Calibration Scale) and γ_{semantic} (Mask Threshold) for LLaDA-8B and Dream-7B respectively. The Softmax Temperature ($\tau = 0.5$) and Layer Stride ($S = 2$) are shared across all experiments.

Benchmark	LLaDA-8B		Dream-7B		Common	
	α	γ	α	γ	τ	Stride
TruthfulQA	2.5	0.5	2.5	0.5	0.5	2
GSM8K	0.5	0.7	0.3	0.1	0.5	2
SVAMP	0.6	0.7	0.7	0.1	0.5	2
Countdown	0.7	0.1	0.8	0.8	0.5	2

A.4 Decoding and Evaluation Protocols

We employ distinct decoding strategies tailored to the downstream task format: **Reasoning Tasks.** For reasoning tasks such as GSM8K, SVAMP, and Countdown, we adopt the **Semi-Autoregressive (SAR) sampling** strategy following the LLaDA protocol (Nie et al., 2025).

- **Block-wise Generation:** The sequence is generated in a left-to-right manner with a maximum generation length of $L_{\text{gen}} = 256$ and a block size of $L_{\text{block}} = 64$ (i.e., 4 blocks per sequence). Within each block, the model performs iterative denoising.
- **Sampling Steps:** We use $T = 128$ total denoising steps for each sequence, distributed evenly across the 4 blocks (i.e., 32 steps per block).
- **Per-Step Mask Ratio:** Since denoising is performed block by block, the gating ratio

Table 6: Open-ended question answering results on TriviaQA.

Method	Accuracy $\uparrow$
LLaDA-8B Vanilla	34.8
LLaDA-8B + TICD	36.5

γ used by TICD is computed within the currently active block: $\gamma = L_{\text{block_mask}}/L_{\text{block}}$, where $L_{\text{block_mask}}$ is the number of masked positions remaining in the current block. This keeps the interpretation of γ consistent with the temporal analysis in Section 3.3: within each block, γ decreases from 1.0 toward 0.0 as denoising progresses.

Multiple-Choice Tasks (TruthfulQA). For discriminative tasks where the goal is to select the most plausible answer from a set of candidates $\mathcal{A} = \{a_1, \dots, a_k\}$, we do not perform generation. Instead, we compute the likelihood of each candidate answer given the question q :

$$\text{Score}(a_i) \approx \mathbb{E}_{t \sim \mathcal{U}(0,T), \epsilon \sim \mathcal{N}(0,I)} [-\mathcal{L}_{\text{diff}}(z_t, t, [q; a_i])] \quad (10)$$

Following LLaDA’s evaluation protocol, we approximate the log-likelihood (ELBO) using Monte Carlo sampling. Specifically, we sample multiple timesteps t and noise instances to estimate the denoising loss $\mathcal{L}_{\text{diff}}$ for the concatenated sequence $[q; a_i]$. The candidate with the highest estimated likelihood (lowest loss) is selected as the prediction.

B TICD Algorithm

We present the full TICD procedure in Algorithm 1. The algorithm switches between vanilla denoising in Regime I and contrastive calibration in Regime II according to the mask ratio γ .

C Additional Experiments

C.1 Open-Ended QA on TriviaQA

We further evaluate TICD on TriviaQA (Joshi et al., 2017) in a 5-shot open-ended QA setting. We use the shared TICD configuration described in Appendix C.3. We report exact-match (EM) accuracy following standard answer normalization. As shown in Table 6, TICD improves the accuracy of LLaDA-8B from 34.8% to 36.5%.

Table 7: Ablation of APC on LLaDA-8B.

Method	GSM8K $\uparrow$	SVAMP $\uparrow$
Vanilla	78.0	84.6
TICD w/o APC	78.4	85.2
TICD (Full)	80.1	87.3

C.2 Ablation of the Adaptive Plausibility Constraint

To examine the contribution of the Adaptive Plausibility Constraint (APC), we evaluate TICD without it using the same configurations as in the main experiments (Appendix A.3). As shown in Table 7, removing APC reduces performance, but the resulting variant still outperforms vanilla decoding on both GSM8K and SVAMP. This suggests that APC is beneficial but is not the sole source of TICD’s gains.

C.3 Shared Configuration

We additionally evaluate TICD using a shared configuration across all tasks and both model backbones. We use candidate layers $\{2, 4, \dots, 30\}$ for LLaDA-8B and $\{2, 4, \dots, 26\}$ for Dream-7B, with $\alpha = 0.6$, $\gamma_{\text{semantic}} = 0.7$, $\tau = 0.5$, and a layer stride of 2. As shown in Table 8, the shared configuration consistently improves over vanilla decoding across all evaluated tasks and models.

D Sensitivity Analysis of Calibration Scale

To further understand the behavior of Time-Aware Internal Contrastive Decoding (TICD), we conduct a sensitivity analysis on the calibration scale parameter α . This hyperparameter controls the strength of the contrastive intervention applied during the inference process. We evaluate the performance of LLaDA-8B on the SVAMP dataset across a range of α values from 0.1 to 1.0.

As illustrated in Figure 6, TICD consistently outperforms the vanilla baseline (84.6%) across a broad range of moderate calibration scales ($\alpha \in [0.1, 0.6]$). The performance reaches its peak at $\alpha = 0.6$, achieving an accuracy of **87.3%**, which demonstrates the effectiveness of our contrastive decoding strategy in enhancing mathematical reasoning.

However, we observe a performance drop when the calibration scale becomes excessive ($\alpha > 0.7$). At $\alpha = 0.8$ and $\alpha = 1.0$, the accuracy falls below the baseline. This phenomenon suggests that while

Algorithm 1 Time-Aware Internal Contrastive Decoding (TICD)

Require: Base dLLM Φ , input tokens $\mathbf{x}$, masked positions $\mathcal{M}$, current mask ratio γ

Require: Candidate layers $\mathcal{L}_{\text{cand}}$, calibration scale α , threshold γ_{semantic} , temperature τ , APC parameter k

Ensure: Calibrated log-probabilities $\tilde{\mathbf{z}}^{(N)}$

```
1:  $\{\mathbf{h}^{(l)}\}_{l=1}^N \leftarrow \Phi.\text{forward\_hidden}(\mathbf{x})$ 
2:  $\mathbf{z}^{(N)} \leftarrow \text{LogSoftmax}(\text{LMHead}(\mathbf{h}^{(N)}))$ 
3: if  $\gamma > \gamma_{\text{semantic}}$  then
4:   return  $\mathbf{z}^{(N)}$  ▷ Regime I: no calibration
5: end if
6:  $\tilde{\mathbf{z}}^{(N)} \leftarrow \mathbf{z}^{(N)}$  ▷ Regime II: apply TICD
7: for all  $i \in \mathcal{M}$  do
8:   for all  $l \in \mathcal{L}_{\text{cand}}$  do
9:      $\mathbf{z}_i^{(l)} \leftarrow \text{LogSoftmax}(\text{LMHead}(\mathbf{h}_i^{(l)}))$ 
10:     $w_{l,i} \leftarrow \text{ComputeWeight}(P_i^{(N)}, P_i^{(l)}, \tau)$  ▷ Eq. (4)
11:  end for
12:   $\Delta \mathbf{z}_i \leftarrow \sum_{l \in \mathcal{L}_{\text{cand}}} w_{l,i} (\mathbf{z}_i^{(N)} - \mathbf{z}_i^{(l)})$ 
13:   $\tilde{\mathbf{z}}_i^{(N)} \leftarrow \mathbf{z}_i^{(N)} + \alpha \cdot \Delta \mathbf{z}_i$ 
14:   $\mathcal{V}_{\text{top}} \leftarrow \text{TopK}(P_i^{(N)}, k)$  ▷ APC
15:   $\tilde{\mathbf{z}}_i^{(N)}[v \notin \mathcal{V}_{\text{top}}] \leftarrow -\infty$ 
16: end for
17: return  $\tilde{\mathbf{z}}^{(N)}$ 
```

Table 8: Results using a shared TICD configuration across all tasks.

Model	Method	TruthfulQA			Reasoning		
		MC1	MC2	MC3	GSM8K	SVAMP	Countdown
LLaDA-8B	Vanilla	31.6	49.4	23.9	78.0	84.6	22.3
	TICD shared	37.8	54.5	27.9	79.9	87.0	25.8
Dream-7B	Vanilla	34.6	53.4	24.5	75.1	81.4	14.8
	TICD shared	36.6	56.7	28.6	76.1	83.3	20.5

moderate calibration can suppress unreliable prediction gaps, an overly large calibration scale may over-amplify final-layer differences and move the distribution away from the base model. Therefore, selecting an appropriate α is crucial for balancing hallucination mitigation and generation quality.

E Visualization Results for Generative Tasks

We provide qualitative comparisons on the GSM8K benchmark across three representative cases in Tables 9, 10, and 11. As observed, the **Baseline** model tends to hallucinate or miss computation steps during multi-hop reasoning. In contrast, **TICD** maintains logical consistency throughout the generation process, leading to the correct numerical answers.

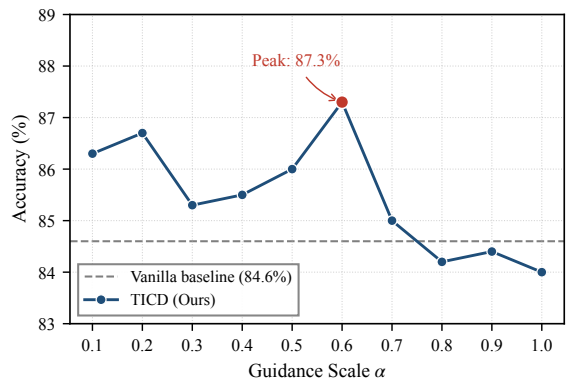


Figure 6: **Sensitivity Analysis of calibration scale (α) on SVAMP.** The dashed blue line represents the vanilla LLaDA-8B baseline (84.6%). TICD achieves optimal performance at $\alpha = 0.6$, while excessive calibration scales ($\alpha > 0.7$) lead to performance degradation.

Table 9: **Qualitative Comparison (Case 1: Logic Scope)**. The **Baseline** model correctly calculates the second week's numbers but fails to aggregate them with the first week, missing the global scope. **TICD** explicitly tracks and sums both periods.

Question	Alani's family decided that the children should write stories of any kind. They were then required to read all of the stories they'd written to the family at the end of the weekend. Alani wrote 20 stories in the first week, her brother Braylen wrote 40 stories, and her sister Margot wrote 60 stories. If they each doubled the number of stories they'd written in the first week in the second week, calculate the total number of stories they wrote altogether
Baseline	To determine the total number of stories written by Alani, Braylen, and Margot, we need to calculate the number of stories each child wrote in both weeks, and then sum these amounts. - Alani wrote 20 stories in the first week and doubled the number in the second week: $20 \times 2 = 40 \text{ stories}$ - Braylen wrote 40 stories in the first week and doubled the number in the second week: $40 \times 2 = 80 \text{ stories}$ - Margot wrote 60 stories in the first week and doubled the number in the second week: $60 \times 2 = 120 \text{ stories}$ Now, we sum the total number of stories written by all three children: $40 + 80 + 120 = 240 \text{ stories}$ <i>(Error: Only summed the second week, ignored the first week)</i> Answer: 240 ✗ Incorrect
TICD (Ours)	To determine the total number of stories written by Alani, Braylen, and Margot, we need to calculate the number of stories each child wrote in both weeks, and then sum them up. First, calculate the number of stories each child wrote in the first week: - Alani: 20, Braylen: 40, Margot: 60 Second, calculate the number of stories each child wrote in the second week by doubling the number of stories they wrote in the first week: - Alani: $20 \times 2 = 40$ - Braylen: $40 \times 2 = 80$ - Margot: $60 \times 2 = 120$ Third, sum the total number of stories written by all three children in both weeks: $\text{Total} = 20 + 40 + 60 + 40 + 80 + 120 = 360$ Answer: 360 ✓ Correct

Table 10: **Qualitative Comparison (Case 2: Unit Mismatch)**. The **Baseline** model erroneously adds a weekly bonus to a monthly wage sum. **TICD** correctly computes the weekly total first, ensuring unit consistency.

Question Watson works a 10-hour shift each day, five days a week. He earns \$10 per hour and gets a \$300 bonus each week if the company performs well. How much money did Watson make in April if the company performed very well for the whole month?

Baseline To determine how much money Watson made in April, we need to follow these steps:

1. Calculate the number of weeks Watson worked in April.
2. Calculate the total number of hours worked in April.
3. Calculate the total earnings from hourly hours.
4. Add the weekly bonus to the total earnings.

First, we know April has four weeks.

Next, we calculate the total number of hours worked in April:

$$50 \text{ hours/week} \times 4 \text{ weeks} = 200 \text{ hours}$$

Then, we calculate the total earnings from hourly hours:

$$200 \times 10 = 2000$$

Finally, we add the weekly bonus to the total earnings:

$$\text{Total} = 2000 + 300 = \$2300$$

(Error: Added a single weekly bonus to 4 weeks of wages)

Answer:

✗ Incorrect

TICD To determine how much money Watson made in April, we need to follow these steps:

- (Ours)**
1. Calculate Watson's hourly earnings.
 2. Determine the weekly earnings, including the weekly bonus.
 3. Calculate the monthly earnings for April.

First, let's calculate Watson's hourly earnings:

He earns \$10 per hour.

Next, let's determine Watson's weekly earnings, including the weekly bonus:

- Wage: $10 \times 5 \times 10 = 500$ per week.
- Bonus: \$300 per week.
- Total Weekly: $500 + 300 = 800$ per week.

Finally, let's calculate Watson's monthly earnings for April:

$$800 \times 4 = 3200$$

Answer:

✓ Correct

Table 11: **Qualitative Comparison (Case 3: Logic Consistency).** The **Baseline** fails to bridge the logical gap between daily and weekly consumption (ignoring the 7 days) and hallucinates the final division. **TICD** correctly expands the time horizon and executes accurate arithmetic.

Question	Each person in a certain household consumes 0.2 kg of rice every meal. Supposing 5 members of the household eat rice every lunch and dinner, how many weeks will a 42 kg bag of rice last?
Baseline	To determine how many weeks a 42 kg bag of rice will last, we need to follow these steps: 1. Calculate the total amount of rice consumed per meal by the entire household. 2. Calculate the total amount of rice consumed per week by the entire household. 3. Determine how many weeks the 42 kg bag of rice will last based on the weekly consumption. First, calculate the total amount of rice consumed per meal: $0.2 \text{ kg/person} \times 5 \text{ members} = 1 \text{ kg/meal}$ Next, calculate the total amount of rice consumed per week: • There are 2 meals in a day. • Total rice per week = $1 \text{ kg/meal} \times 2 \text{ meals/week} = 2 \text{ kg/week}$ (Logic Error: Failed to multiply by 7 days. Mistook daily sum for weekly sum.) Finally, determine how many weeks the 42 kg bag of rice will last: $42 \text{ kg} / 2 \text{ kg/week} = 7 \text{ weeks}$ (Arithmetic Error: Hallucination, $42/2 \neq 7$) Answer: 7 ✗ Incorrect
TICD (Ours)	To determine how many weeks a 42 kg bag of rice will last, we need to calculate these steps: 1. Calculate the total amount of rice consumed per meal by the household. 2. Calculate the total amount of rice consumed per week by the household. 3. Determine how many weeks the 42 kg bag of rice will last. First, calculate the total amount of rice consumed per meal: • Total rice consumed per meal = $0.2 \text{ kg/person} \times 5 \text{ members} = 1 \text{ kg/meal}$ Next, calculate the total amount of rice consumed per week: • There are 2 meals per day (lunch and dinner). • Calculation: $1 \text{ kg/meal} \times 2 \text{ meals/day} \times 7 \text{ days/week} = 14 \text{ kg/week}$ Finally, determine how many weeks the 42 kg bag of rice will last: $\text{Weeks} = 42 \text{ kg} / 14 \text{ kg/week} = 3 \text{ weeks}$ Answer: 3 ✓ Correct